www.farelettronica.com

fare elettronica EXTRA
n. 1 DICEMBRE 2008 - € 6,00

AVR projects

volume 1

una raccolta di progetti con i prestigiosi microcontrollori ATMEL

i microcontrollori

- ATtiny12
- ATtiny13
- ATmega8
- ATmega16
- ATmega32
- AVRMmega168
- AT90S2313

Rilevatore di gas

Orologio via radio in DCF77

Interruttore touch

FAN controller

Uso della Compact Flash

Una sveglia LCD

Misuratore di distanza ad ultrasuoni

Termometro multicanale

Generatore di barre PAL

Contagiri

Datalogger GPS

Light Dimmer per moto

Display a testo scorrevole

...e molti altri ancora!

EXTRA

Microcontrollers

Digital Signal Controllers

Analog

Serial EEPROMs

Rilevamento tattile capacitivo in una soluzione flessibile a singolo chip

Le interfacce tattili capacitive offrono un mezzo eccellente per arricchire il vostro progetto con nuove soluzioni di comando seducenti, a basso costo ed affidabili. La Sensing Solution mTouch™ di Microchip Technology comprende tutti i kit di sviluppo e tutti gli strumenti diagnostici gratuiti necessari per rendere l'implementazione semplice e rapida. Il nostro codice sorgente gratuito può essere integrato in modo ottimizzato nel firmware esistente, inserendolo così nel contesto di un unico microcontroller PIC® ed evitando il ricorso a controller aggiuntivi.

La Sensing Solution mTouch prevede:

- Licenza per librerie e codice sorgente GRATUITI
- Strumenti di diagnosi GRATUITI
- Integrazione con i microcontroller PIC a 8- e 16-bit
- Facilità di espansione: supporto da 6 a 100 pin
- Funzionamento a basso consumo

3 SEMPLICI PASSI PER INIZIARE

1. Visitate il design center mTouch Sensing Solutions al link www.microchip.com/mTouch
2. Scaricate gratuitamente le librerie e il codice sorgente
3. Per un periodo di tempo limitato, acquistando da www.microchipDIRECT.com e utilizzando il codice coupon **EUMTOUCH**, potrete risparmiare il 20% su un'ampia varietà di tool di sviluppo per rilevamento tattile

Intelligent Electronics start with Microchip

www.microchip.com/mTouch

14
Dimmer per moto
Con questo circuito potrete controllare la luminosità di una o due lampade a 6/12V. Un gadget ideale per la vostra moto.

16
Gestione memorie Compact Flash
Ecco un semplice schema e le relative routine software per gestire una memoria CF con un AT90S2313.

18
Orologio DCF77 con LCD
Questo progetto impiega un ricevitore per ricevere il segnale orario dalla stazione tedesca DCF77 e lo visualizza su un display LCD alfanumerico HD44780 compatibile.

20
FAN controller
Il modo più semplice per variare la velocità di rotazione di una ventola in funzione della temperatura.

22
GAS detector
Un rilevatore di gas ad allarme acustico.

26
GPS datalogger
Un circuito in grado di memorizzare le coordinate geografiche ricevute da un GPS per poi renderle disponibili su un PC attraverso una connessione RS232.

30
Termometro multicanale
Un circuito in grado di monitorare la temperatura in punti diversi impiegando fino a 16 sensori one-wire DS1820.

38
Generatore di barre video
Con un AT90S2313 e pochi altri componenti aggiuntivi potrete realizzare questo semplicissimo generatore di barre video in standard PAL.

42
Contagiri
Ecco un circuito per realizzare un contagiri elettronico basato su un sensore ad infrarossi.

44
Sveglia con LCD grafico
Un orologio realizzato con un ATmega32 ed un display LCD grafico. La particolarità? La gestione intelligente della retroilluminazione del display.

48
Testo scorrevole
Con questo semplice progetto potrete visualizzare dei testi scorrevoli su una matrice di LED 15x7.

50
Touch Switch
Un interruttore tattile il cui funzionamento è basato sui tempi di carica e scarica di un condensatore autocostruito.

52
Metro ad ultrasuoni
Un interessante progetto per la misurazione delle distanze sfruttando l'eco di un emettitore ad ultrasuoni.

54
Monitor per batterie
Questo circuito effettua la scarica di una batteria monitorando tutti i parametri in modo da ricostruire la curva caratteristica e la resistenza interna.

56
Lampada da campeggio a LED
una modifica ad una lampada da campeggio commerciale per sostituire la lampada ad incandescenza con una più efficiente a LED della quale è possibile variare anche il grado di luminosità.

62
AVR touchscreen
Gestire un display grafico touchscreen con un ATmega16. L'hardware ed il software.

EXTRA *info*

6 Sviluppare con gli AVR
Una guida allo sviluppo di applicazioni con i microcontrollori AVR destinata ai principianti e a tutti coloro che si avvicinano per la prima volta al mondo dei microcontrollori Atmel.

34 AVR Instruction set
Il set completo di istruzioni assembler per i microcontrollori AVR.

40 AVR pinout
Le pedinature dei microcontrollori AVR utilizzati in questo volume.

a cura di MAURIZIO DEL CORSO

indice per AVR

AT90S2313
GESTIONE MEMORIE COMPACT FLASH *pag. 16*
GENERATORE DI BARRE VIDEO *pag. 38*
CONTAGIRI *pag. 42*
METRO AD ULTRASUONI *pag. 52*

ATmega128
GPS DATALOGGER *pag. 26*

ATmega16
AVR TOUCHSCREEN *pag. 62*

ATmega168
TERMOMETRO MULTICANALE *pag. 30*

ATmega8
SVEGLIA CON LCD GRAFICO *pag. 44*
DIMMER PER MOTO *pag. 14*
GAS DETECTOR *pag. 22*
TESTO SCORREVOLE *pag. 48*

ATmega88
MONITOR PER BATTERIE *pag. 54*

ATtiny12
OROLOGIO DCF77 CON LCD *pag. 18*

ATtiny13
TOUCH SWITCH *pag. 50*

ATtiny15
FAN CONTROLLER *pag. 20*
LAMPADA DA CAMPEGGIO A LED *pag. 56*

indice per compilatore

AVR Assembler
METRO AD ULTRASUONI *pag. 52*
OROLOGIO DCF77 CON LCD *pag. 18*
LAMPADA DA CAMPEGGIO A LED *pag. 56*

AVR Studio 4
GESTIONE MEMORIE COMPACT FLASH *pag. 16*
GENERATORE DI BARRE VIDEO *pag. 38*
TERMOMETRO MULTICANALE *pag. 30*
TOUCH SWITCH *pag. 50*
FAN CONTROLLER *pag. 20*

Bascom AVR
MONITOR PER BATTERIE *pag. 54*

Codevision
CONTAGIRI *pag. 42*
GAS DETECTOR *pag. 22*
TESTO SCORREVOLE *pag. 48*

MikroC AVR
AVR TOUCHSCREEN *pag. 62*

WinAVR
GPS DATALOGGER *pag. 26*
SVEGLIA CON LCD GRAFICO *pag. 44*
DIMMER PER MOTO *pag. 14*

AVR per tutti

Dopo il successo dei due fascicoli speciali dedicati ai progetti con i PIC abbiamo deciso di dedicare spazio ai microcontrollori AVR spinti anche dalle numerose richieste pervenuteci in redazione. Ecco dunque il primo volume di "AVR Projects", una raccolta di progetti tutti rigorosamente basati sui microcontrollori Atmel. E se non siete esperti di AVR non disperate! Nelle pagine di "AVR projects" troverete anche un tutorial per iniziare a sviluppare con questi micro, le pedinature dei micro utilizzati ed il set completo di istruzioni assembly comuni a tutti gli AVR. Non vi resta che dilettarvi nella realizzazione dei progetti e nella loro personalizzazione grazie anche alla disponibilità dei file sorgente che sono scaricabili gratuitamente dal sito www.farelettronica.com/avr.

SVILUPPARE CON GLI

Una guida allo sviluppo di applicazioni con i microcontrollori AVR destinata ai principianti e a tutti coloro che si avvicinano per la prima volta al mondo dei microcontrollori Atmel

Il compito più difficile iniziando lo sviluppo con nuovi microcontrollori, sembra essere quello di procurarsi le informazioni e la documentazione necessarie per riuscire a far funzionare un primo programma per AVR. In questo tutorial introduttivo si assume che non abbiate alcuna conoscenza dell'architettura o del set d'istruzioni degli AVR. Tutto ciò di cui avrete bisogno per completare il tutorial è un PC dotato di sistema operativo Windows e una buona dose di pazienza.

Gli strumenti necessari

Facciamo una partenza facilitata scaricando i file di cui avremo bisogno nel seguito. Per prima cosa dovreste scaricare i file dell'esempio da www.farelettronica.com/avr in modo da averli a disposizione quando necessario. Scaricate i file in una directory temporanea (ad es. C:\Temp). Abbiamo ora bisogno dell'ambiente di sviluppo AVR studio che potrete scaricare dal sito www.atmel.com/avr. AVR studio comprende un editor ed un simulatore che useremo per scrivere il nostro codice e per vedere come viene eseguito da un dispositivo AVR. Il micro utilizzato in questo tutorial è l'AT90S8515 un dispositivo particolarmente adatto per iniziare con gli AVR.

Per adesso non preoccupatevi riguardo ai differenti tipi di micro AVR. Vedrete che sono molto simili fra loro, e una volta imparato ad usarne uno (come l'8515) sarete in grado di usare qualsiasi altro AVR senza problemi. Il datasheet ed il set di istruzioni sono disponibili per il dawnload su www.farelettronica.com/avr.

Installazione di AVR Studio 4

Per gli utenti di Windows NT/2000/XP è necessario connettersi con diritti di amministratore per poter installare con successo AVR Studio. Una volta scaricati i file di AVR studio, iniziate l'installazione facendo doppio click sul file AVRSTUDIO.EXE. Si tratta di un file autoestraente: vi verrà quindi chiesto dove volete estrarre i file. Il percorso di default punta al vostro folder temporaneo di default, che potrebbe essere ben "nascosto" nel vostro hard disk, così assicuratevi di ricordarvelo, o scegliete un nuovo percorso nel quale estrarre i file (ad es. c:\temp). Una volta che i file sono stati estratti, apri-

Figura 1: la finestra di avvio di AVR Studio4.

te la directory temporanea, e fate doppio click sul file SETUP.EXE. Proseguite con l'installazione, utilizzando il path d'installazione di default. NB: potete usare un altro percorso, ma questo tutorial prevede che abbiate installato AVR Studio nel percorso di default.
Questo è tutto. Adesso avete installato tutto il software di cui avete bisogno per scrivere codice ed eseguire programmi per tutti i dispositivi AVR disponibili! Conservate il Datasheet e l'Instruction Set Manual in un posto da possiate ricordare.

Figura 2: configurazione delle Impostazioni del Progetto.

Introduzione agli AVR

I microcontrollori AVR sono suddivisi i tre gruppi:

1. tinyAVR
2. AVR (Classic AVR)
3. mega AVR

La differenza tra questi dispositivi consiste nelle diverse caratteristiche disponibili. I tinyAVR sono di solito dispositivi con meno pin ed una dotazione ridotta se confrontati con i megaAVR. Tutti i dispositivi AVR possiedono lo stesso set d'istruzioni e la stessa organizzazione della memoria, così da facilitare la migrazione da un dispositivo AVR all'altro. Alcuni AVR comprendono SRAM, EEPROM, un'interfaccia per SRAM esterna, convertitori Analogico/Digitale, moltiplicatore hardware, UART, USART e così via. Se prendete un tinyAVR ed un megaAVR e togliete tutti i moduli periferici, rimarrete con il solo Core AVR. Questo Core è lo stesso per tutti i dispositivi AVR. La morale è che le denominazioni tinyAVR, AVR (Classic AVR) e megaAVR non indicano diverse performance, ma sono più un'indicazione della "complessità" del dispositivo: moltissime dotazioni per i megaAVR, un insieme ridotto per i tinyAVR. Gli AVR (Classic AVR) sono in qualche modo una via di mezzo, mentre le distinzioni tra i vari gruppi diventano sempre più vaghe. Così se per il vostro progetto avete un budget limitato dovreste scegliere un AVR che comprenda le sole dotazioni di cui avete bisogno, altrimenti potrete optare per il modello più grande possibile in modo da non avere alcuna limitazione.

Figura 3: elezione della Piattaforma di Debug.

Sebbene sia pienamente possibile imparare a programmare gli AVR con la sola lettura del datasheet, questo è un approccio complesso e lento. In questo tutorial useremo un approccio facile e veloce, che consiste in:

• Trovare del codice già scritto e funzionante.
• Capire come funziona tale codice
• Modificarlo per adattarlo ai nostri scopi.

Leggere i Datasheet degli AVR

È facile impaurirsi dando uno sguardo ai Datasheet degli AVR. Ad esempio, il datasheet degli ATmega128(L) è lungo circa 350 pagine, e leggerlo dall'inizio alla fine-riuscendo a ricordarne il contenuto-è un compito notevole. Fortunatamente non si presuppone che facciate né l'una né l'altra cosa. I datasheet sono dei documenti tecnici completi che dovreste usare come riferimento nel momento in cui foste in dubbio su come funziona una data periferica o caratteristica. Quando aprite un datasheet AVR potete notare che può essere suddiviso nei gruppi seguenti:

1. La prima pagina contenente le informazioni principali e l'elenco delle caratteristiche.
2. Panoramica delle caratteristiche (Architectural Overview).
3. Descrizione delle Periferiche.
4. Programmazione della Memoria.

LISTATO 1 - SAMPLECODE.ASM

```
;My Very First AVR Project
.include "8515def.inc"      ;Include il file di definizione dell'8515
.def Temp = R16             ;Assegna al Registro R16 il nome Temp
.org 0x0000                 ;Il codice seguente viene posto a partire dalla
                            ;locazione 0x0000
    rjmp RESET              ;Salta (relative Jump) all'etichetta RESET
RESET:               ;Etichetta Reset
    ldi Temp, 0xFF          ;Copia 255 in R16 (abbiamo definito
                            ;R16 = Temp)
    out DDRB, Temp          ;Copia questo valore nel registro Data
                            ;direction di PORTB
Loop:                       ;Etichetta Loop
    out PORTB, Temp  ;Pone alti tutti i bit (255 decimale) di PORTB
    dec Temp                ;Decrementa R16 (Temp)
    rjmp Loop        ;Salta all'etichetta Loop
```

Figura 4: la vostra prima linea di codice.

5. Caratteristiche.
6. Sommario dei Registri.
7. Sommario del Set d'Istruzioni (Instruction Set Summary).
8. Informazioni sui package.
Ciò è decisamente comodo. Quando avrete familiarizzato con l'uso del datasheet dell'AT90S8515, migrare ad un altro datasheet sarà un gioco da ragazzi.
Dopo aver completato questo tutorial dovreste spendere un po' di tempo per leggere le sezioni dell'Architectural Overview dei datasheet (poste all'inizio dei datasheets). Queste sezioni contengono un sacco di informazioni utili sulle memorie degli AVR, sui modi d'indirizzamento ed altre utili informazioni. Un'altra pagina utile da vedere è il sommario del set d'istruzioni. Si tratta di un riferimento che vi farà comodo quando inizierete a sviluppare del codice vostro. Se poi volete informazioni approfondite su un'istruzione, potete semplicemente cercarla nell'Instruction Set Manual che avete scaricato in precedenza! OK! Ora avete installato il software, possedete una vaga conoscenza dei diversi tipi di AVR, e sapete che esistono un sacco di informazioni nel datasheet di cui non sapete ancora assolutamente nulla! Bene, ora è venuto il momento di cominciare a sviluppare!

Usare AVR Studio 4

Passo 1:
Creazione di un nuovo Progetto
Lanciate AVR Studio lanciando AVR Studio 4 da [start] | [Programmi] | [Atmel AVR Tools]. AVR Studio si avvierà, presentando la finestra di dialogo mostrata in **figura 1**.

Passo 2:
Configurazione delle Impostazioni del Progetto
A questo punto si impostano il tipo di progetto e si scelgono i nomi dei file e le directory dove devono essere memorizzati **(figura 2)**.
Questo è realizzato in quattro passi:
1. Cliccare su 1 per indicare che volete creare un programma in Assembly.
2. Il nome del progetto può essere un nome qualsiasi, ma "Leds" è indicativo di quello che farà il nostro programma.
3. Specificare se AVR Studio deve creare un file assembly automaticamente. È quello che vogliamo. Il nome del file può essere uno qualsiasi, ma usate "Leds" per rimanere compatibili con questo tutorial!
4. Selezionate il percorso dove volete memorizzare i vostri file.
5. Verificate ogni cosa ancora una volta, ed accertatevi che entrambe le caselle siano spuntate. Una volta soddisfatti, premete il bottone "Next >>".

Passo 3:

Selezione della Piattaforma di Debug

Il software AVR Studio 4 può essere usato come front-end per una vasta gamma di tool di debug. AVR Studio supporta una vasta gamma di tool per l'emulazione ed il debug.

Dato che ancora non abbiamo acquistato nessuno di questi, useremo il simulatore incorporato.

Selezioniamo il dispositivo AT90S8515.

Verifichiamo le nostre impostazioni ancora una volta, quindi premiamo "Finish" per creare il progetto ed andare al file assembly.

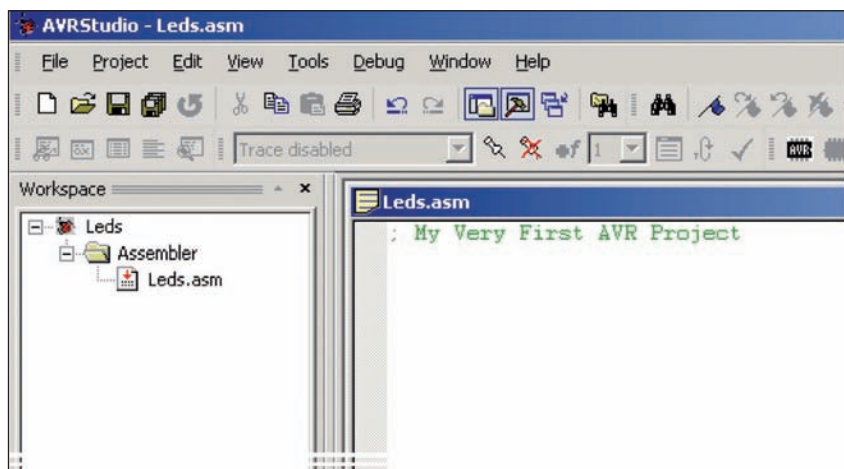



Figura 5: il file leds.asm è stato salvato.

Passo 4:

Scrivere la vostra prima linea di codice

AVR Studio partirà ed aprirà un file vuoto denominato Leds.asm. Daremo uno sguardo più da vicino alla GUI di AVR Studio più avanti. Per adesso notate che Leds.asm non è elencato nel folder "Assembler " nella colonna di sinistra. Ciò perché il file non è ancora stato salvato. Scrivete questa riga: "; My Very First AVR Project" come mostrato nella **figura 4**. Il punto e virgola ';' indica che il resto della riga deve essere considerato un commento dall'assembler.

Per salvare la riga premette <CTRL> - S o selezionate [Save] dal menu [File]. Il file Leds.asm comparirà allora nella colonna di sinistra come mostrato in **figura 5**.

La GUI di AVR Studio 4

Diamo uno sguardo più da vicino alla Graphical User Interface (GUI) di AVR Studio riportata in **figura 6**. Come potete vedere, abbiamo suddiviso la GUI in 6 sezioni. AVR Studio 4 è dotato di un sistema di help per AVR Studio, per cui spiegheremo semplicemente la struttura generale di AVR Studio 4 riferendo dove potete trovare informazioni approfondite tramite l'help on-line di AVR Studio.

1. La prima riga è il "menu". Qui troverete i menu standard di windows come save e load file, Cut & Paste, ed altri menu specifici di AVR Studio, come Emulation options ed altro.

2. Le righe successive sono delle Toolbar, che rappresentano delle scorciatoie per le funzioni più comuni. Queste funzioni possono servire per salvare file, aprire nuove finestre, settare breakpoint e così via.

3. Il Workspace contiene delle informazioni sui file del vostro progetto, la vista IO, ed informazioni riguardanti il dispositivo AVR prescelto.

4. Questa è la finestra dell'editor. Qui potete scrivere il vostro codice assembly. È anche possibile integrare in AVR Studio un compilatore C, ma questo è argomento per utenti più avanzati.

LISTATO 2 - CODICE D'ESEMPIO

```
; My Very First AVR Project
.include "8515def.inc"        ;Include il file di definizione dell'8515
.def Temp = R16               ;Assegna al Registro R16 il nome Temp
.org 0x0000                   ;Il codice seguente viene posto a partire dalla
                              ;locazione 0x0000
    rjmp RESET                ;Salta (relative Jump) all'etichetta RESET
RESET:                 ;Etichetta Reset
    ldi Temp, 0xFF            ;Copia 255 in R16 (abbiamo definito
                              ;R16 = Temp)
    out DDRB, Temp            ;Copia questo valore nel registro Data
                              ;direction di PORTB
Loop:                         ;Etichetta Loop
    out PORTB, Temp    ;Pone alti tutti i bit (255 decimale) di PORTB
    dec Temp                  ;Decrementa R16 (Temp)
    rjmp Loop          ;Salta  all'etichetta Loop

; My Very First AVR Project
```

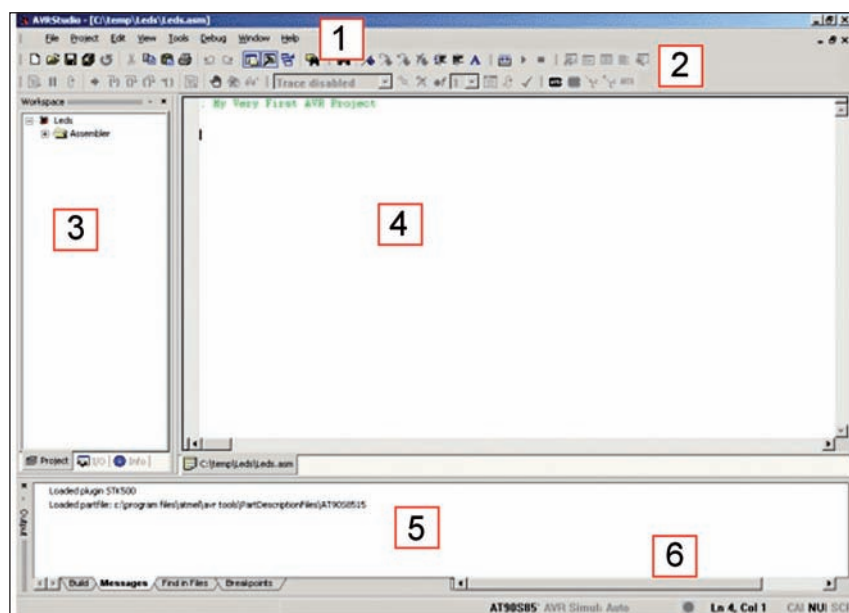



Figura 6: la Graphical User Interface (GUI) di AVR Studio.

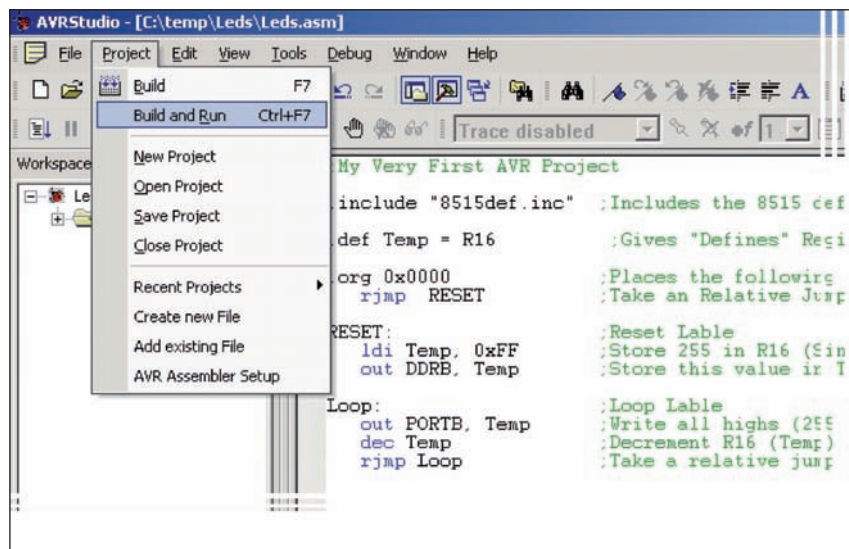



Figura 7: una volta editato il file sorgente passate alla compilazione cliccando su [Build and Run] dal menu [Project].

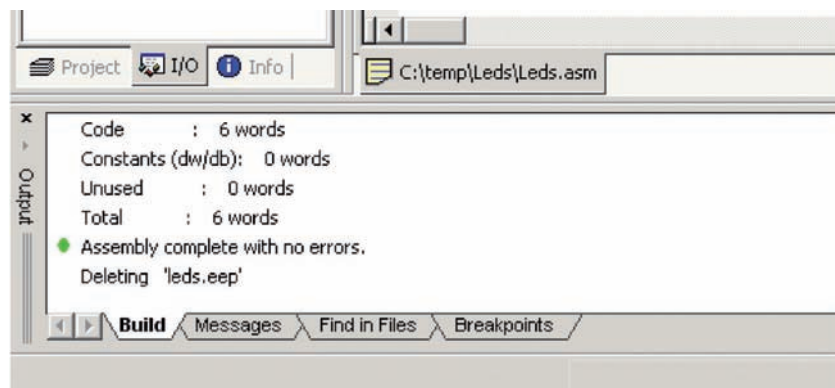



Figura 8: la finestra di output indica che il progetto è stato compilato correttamente senza alcun errore!

5. Output Window. Le informazioni sullo stato sono visualizzate qui.
6. La barra di stato (System Tray) visualizza informazioni riguardo alla modalità d'esecuzione di AVR Studio. Dato che stiamo usando l'AT90S8515 nella modalità simulazione, sarà visualizzato questo.

Il vostro primo programma per AVR

Nell'editor di AVR Studio, proseguite con il vostro programma aggiungendo il testo riportato nel **listato 1** e che troverete nei download relativi a questo articolo su www.farelettronica.com/avr. Notate che il codice sorgente cambia colore nella finestra dell'editor. Questa funzionalità è denominata syntax highlighting ed è molto utile per rendere il codice più leggibile. Una volta inserito il codice sorgente, premete CTRL + F7 o selezionate [Build and Run] dal menu [Project] **(figura 7)**. Nella vista output (nella parte inferiore sinistra dello schermo) dovreste ottenere l'output riportato in figura 8 che indica che il progetto è stato compilato correttamente senza alcun errore! In questa finestra di output possiamo anche vedere che il nostro programma consiste di 6 word di codice (12 bytes). Se non riuscite a compilare il programma, controllate che non ci siano errori di battitura nel file assembly. Se avete posto i file di include (8515def.inc) in un folder diverso da quello di default, dovete inserire il percorso completo nel comando:

```
.include <percorso completo>\8515def.inc.
```

Capire il Codice Sorgente

A questo punto il codice è stato compilato senza errori. Vediamo ora in dettaglio cosa fa il programma appena compilato e farci un'idea di come dovremmo simulare il codice per verificare che si comporti realmente nella maniera voluta. Il **listato 2** mostra il codice sorgente completo.Le linee che iniziano con il punto e virgola sono dei commenti. I commenti possono essere aggiunti a qualsiasi linea di codice. Se i commenti devono occupare più

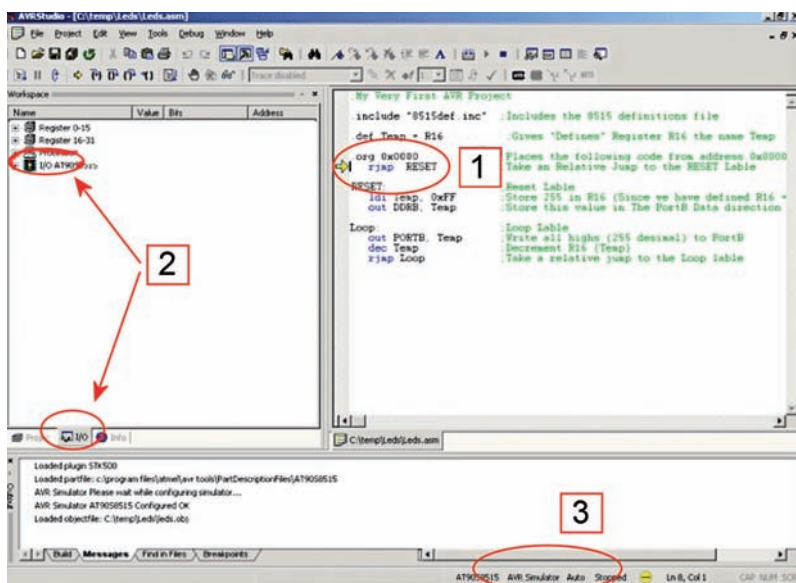



Figura 9: la modalità debugging.

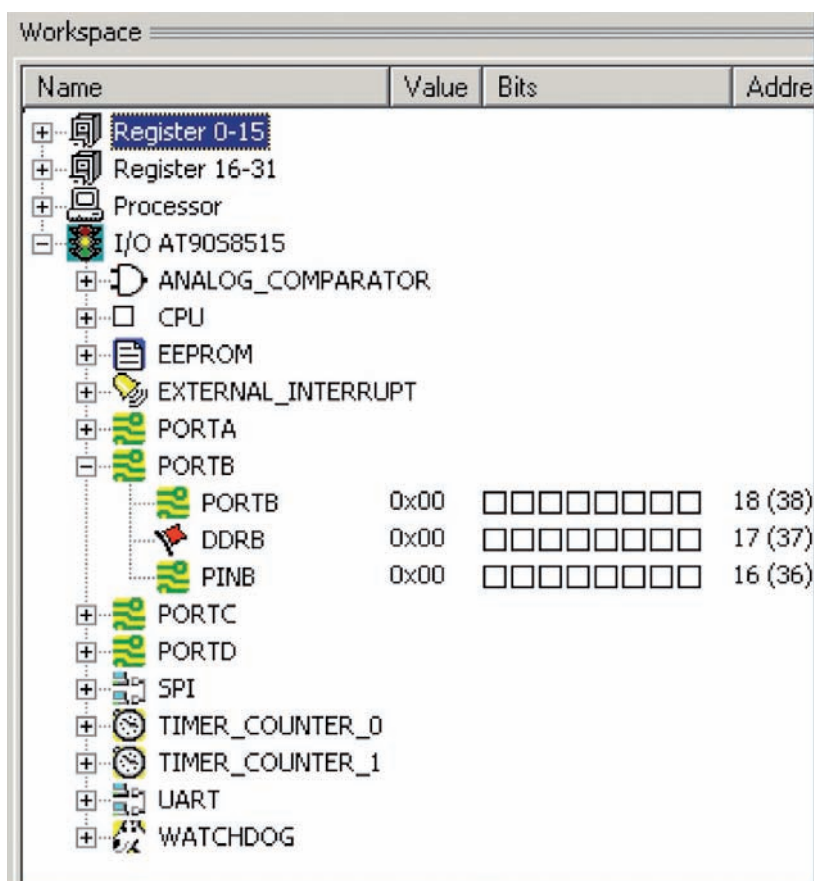



Figura 10: la IO view espansa per una migliore visualizzazione.

linee, ogni linea deve iniziare con un punto e virgola.

.include "8515def.inc"

Diversi dispositivi AVR hanno ad esempio il registro PORTB posizionato in diverse locazioni nella memoria IO. Questi file .inc fanno corrispondere i MNEMONICI agli indirizzi fisici. Ciò ci permette ad esempio di usare l'etichetta PORTB invece di ricordarci la locazione fisica nella memoria di IO (0x18 per l'AT90S8515).

.def Temp = R16

La .def (Define) ci permette di creare etichette (labels) facili da ricordare (ad es. Temp) invece di usare il nome di default del registro (ad es. R16). Ciò è particolarmente utile in progetti nei quali si usano molte variabili mamorizzate nei registri di uso generale (General Purpose registers).

.org 0x0000

Questa è una direttiva dell'assembler che indica al compilatore di posizionare il codice che segue a partire dalla locazione 0x0000 nella memoria Flash. Facciamo questo per fare in modo che l'istruzione successiva RJMP sia posizionata nella locazione 0 (la prima locazione della Flash). La ragione di ciò è che questa locazione è il vettore di Reset (Reset Vector), la locazione dalla quale inizia l'esecuzione di un programma dopo un reset hardware, il power-on o il Watchdog reset. Qui ci sono anche altri vettori di interrupt, ma la nostra applicazione non fa uso di interrupt, così possiamo usare questo spazio per del codice normale!

rjmp RESET

Dato che il comando precedente era .org 0x0000, questa istruzione di salto relativo è posizionata nella locazione 0 della memoria Flash, ed è la prima istruzione ad essere eseguita. Se controllate l'Instruction Set Summary, vedrete che l'AT90S8515 non ha un'istruzione JMP. Ha solo l'istruzione RJMP! Il motivo di ciò è che non avete bisogno dell'istruzione JMP. Se confrontate la JMP e la RJMP vedrete che l'istruzione JMP ha un range maggiore, ma richiede una word d'istruzione in più, rendendo il codice più lento e di maggiori dimensioni. RJMP può raggiungere tutto l'array della memoria Flash dell'AT90S8115, così che l'istruzione JMP non è necessaria, per cui non è implementata.

RESET:

Questa è un'etichetta. Potete porre un'etichetta dove volete nel codice, ed usare le diverse istruzioni di salto per saltare a questa locazione. Questo è davvero comodo, dato che l'assembler stesso calcola l'indirizzo cor-

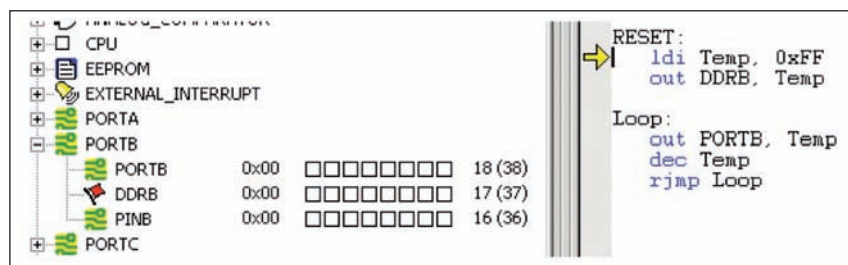



Figura 11: il single-stepping del codice. Inizio della simulazione.

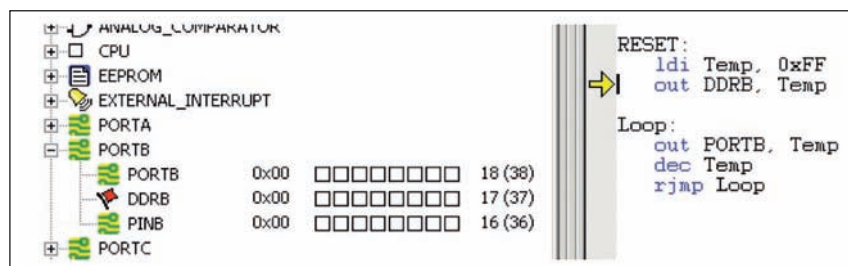



Figura 12: il single-stepping del codice. Esecuzione della seconda istruzione.

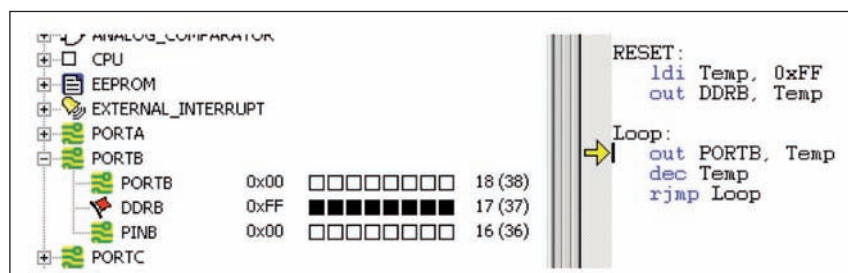



Figura 13: il single-stepping del codice. La IO view mostra DDRB al valore 0xFF.

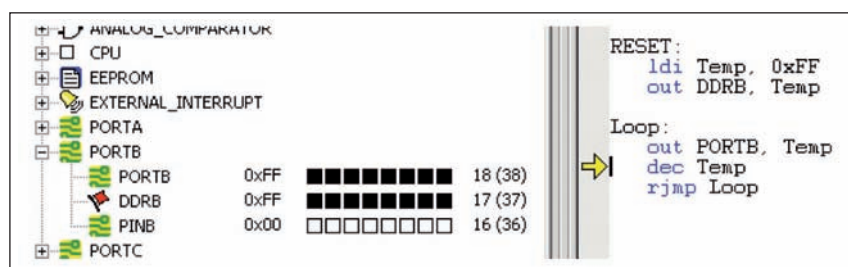



Figura 14: esecuzione delle istruzioni successive in single-stepping.

retto dell'etichetta. Come etichetta potete usare qualsiasi parola vogliate.

ldi Temp, 0xFF

Load Immediate (LDI). Questa istruzione carica un valore immediato, e lo scrive nel registro indicato. Dato che abbiamo definito il registro R16 come "Temp", questa istruzione nel nostro caso scriverà il valore esadecimale 0xff (255 decimale) nel registro R16.

out DDRB, Temp

Perchè non scriviamo semplicemente "ldi DDRB, Temp"? Una buona domanda, la cui risposta richiede uno sguardo all'Instruction Set Manual. Guardate le istruzioni "LDI" ed "OUT". Troverete che LDI ha la sintassi: "LDI Rd, k" che significa che può essere usata solo con i registri General Purpose da R16 ad R31. Guardando l'istruzione "OUT" vediamo che la sua sintassi è "OUT A, Rr" che significa che il contenuto che sarà scritto dall'istruzione OUT andrà ricavato da uno dei 32 (da R0 ad R31) registri General Purpose. Ad ogni modo, questa istruzione pone alti tutti i bit del registro Data Direction di PORTB (DDRB). Ponendo questo registro a 0xFF, tutti i pin di IO sulla Porta B saranno configurati come uscite.

Loop:

Un'altra etichetta...

out PORTB, Temp

Scrive il valore 0xFF in PORTB, cosa che porrebbe i pin di IO di PORTB a 5V (Vcc) se noi li controllassimo su un dispositivo reale. Poiché le porte di IO sono forse la dotazione più usata degli AVR, sarebbe una buona idea guardare il Datasheet riguardo a PORTB. Notate che PORTB ha 3 registri: PORTB, PINB e DDRB.

Nel registro PORTB scriviamo il valore che vogliamo porre sui pin fisici di IO. Nel registro PINB possiamo leggere il valore logico che è attualmente presente sui pin fisici di IO, ed il registro DDRB determina se un dato pin di IO dev'essere configurato come ingresso o uscita (la ragione per cui ci sono 3 registri è il problema del "Read-Modify-Write" associato al più comune utilizzo di 2 soli registri, ma questo è un argomento per un corso avanzato).

dec Temp

L'istruzione Decrement (DEC) decrementa il registro Temp (R16). Dopo che questa istruzione è eseguita, il contenuto di Temp diventa 0xFE. Si tratta di un'istruzione aritmetica, e l'AVR ha un vasto insieme di istruzioni aritmetiche. Per un elenco completo delle istruzioni disponibili fate riferimento all'Instruction Set Summary del datasheet!

rjmp Loop

Qui facciamo un salto all'indietro verso l'etichetta Loop. Il programma continuerà quindi a scrivere la variabile Temp nel registro PORTB decrementadola ad ogni ciclo.

Con questo programma abbiamo realizzato un contatore che conta all'indietro da 255 a 0, ma cosa succede quando raggiungiamo lo zero? Per scoprirlo, simuleremo il comportamento del programma in AVR Studio 4.

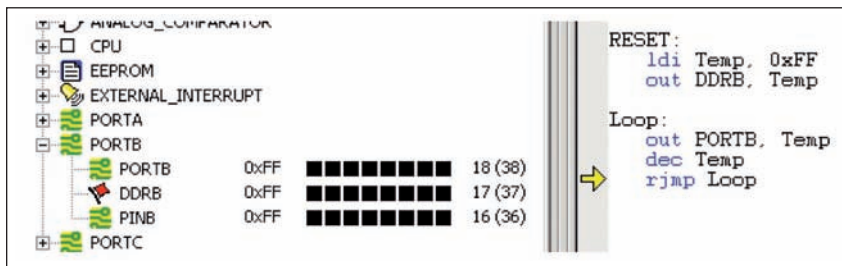



Figura 15: esecuzione delle istruzioni successive in single-stepping.

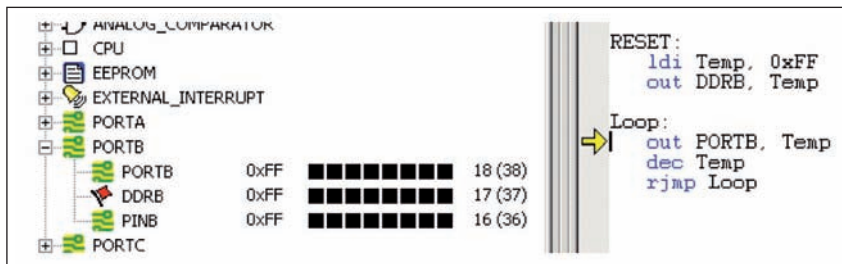



Figura 16: esecuzione delle istruzioni successive in single-stepping.

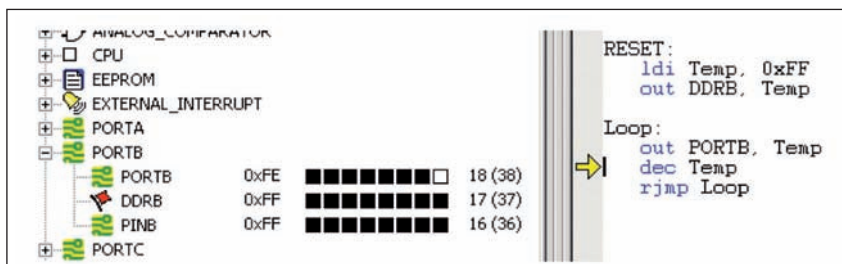



Figura 17: esecuzione delle istruzioni successive in single-stepping.

Simulare il Codice Sorgente

AVR Studio 4 può operare in differenti "modalità". In precedenza quando stavamo scrivendo il codice, eravamo nella modalità editor, ora siamo nella modalità debugging. Diamo uno sguardo più da vicino a questa modalità.

1.Notate che è comparsa una freccia gialla sulla prima istruzione RJMP. Questa freccia indica l'istruzione che sta per essere eseguita.

2.Notate che il workspace è cambiato da Project a IO view. La IO view è il nostro "buco nella serratura" per guardare all'interno dell'AVR, e sarà probabilmente la view che userete maggiormente.

3. La linea in basso contiene le informazioni di stato. Leggiamo: AT90S8515, Simulator, Auto, Stopped. Questa è seguita da un'icona gialla. È buona norma controllare queste informazioni per verificare che abbiate selezionato il dispositivo ed il tool di emulazione corretti.

Dato che il nostro programma opera sui registri della PORTB, espanderemo la IO view in modo da dare uno sguardo più da vicino al contenuto di questi registri. Espandete la IO view (l'albero) come mostrato nella **figura 10**. AVR Studio permette di eseguire il codice alla velocità massima fino ad un certo punto per poi fermarsi. Possiamo comunque rendere l'esecuzione più lenta, premendo manualmente un bottone per ogni istruzione che deve essere eseguita. Questo è detto single-stepping del codice.

Premete [F11] una volta. Questo è il tasto per il single-stepping. Notate che la freccia gialla punta all'istruzione LDI Temp, 0xFF. Questa è la prossima istruzione che sarà eseguita.

Premete [F11] ancora una volta. L'istruzione LDI viene eseguita, e la freccia punta all'istruzione OUT. Il registro Temp ha ora il valore 0xFF (se aprite l'albero "Register 16-31" vedrete che R16 contiene 0xFF: abbiamo definito Temp corrispondente ad R16).

Premete [F11]. DDRB è ora 0xFF, come mostrato nella IO view qui sopra ciò è rappresentato da quadratini neri. Così un quadratino bianco rappresenta il valore logico "0" ed i quadratini neri rappresentano il valore logico "1". Ponendo DDRB tutto alto, tutti i bit di PORTB sono configurati come uscite.

Premete [F11]. 0xFF viene ora scritto nel registro PORTB, e la freccia punta all'istruzione DEC. Notate che PORTB è pari a 0xFF. Notate anche che il registro PINB è ancora 0x00!

Premete [F11]. La variabile Temp è decrementata (0xFF-1 = 0xFE). Inoltre il registro PINB cambia da 0x00 a 0xFF! Perché? Per scoprire perché avviene ciò guardate la sezione PORT del datasheet. La spiegazione è che PORTB è dapprima emesso sul pin, quindi riportato nel registro PIN dandoci un ritardo pari ad un ciclo di clock. Come potete vedere, il simulatore si comporta come il dispositivo reale! L'istruzione successiva è il salto relativo che ci riporta all'etichetta Loop.

Premete [F11]. Viene eseguita l'istruzione RJMP, e la freccia punta di nuovo all'istruzione OUT PORTB, Temp.

Premete [F11] per scrivere il nuovo valore di Temp nel registro PORTB. Notate che il contenuto di PORTB adesso è aggiornato a 0xFE!

Continuate a premere [F11] finché non abbiate portato il registro PORTB a 0x00. Cosa succede se continuate ad eseguire il programma?

Conclusioni

Dopo aver scorso questa introduzione dovreste avere un'idea di base su come impostare ed eseguire un programma sugli AVR. Come menzionato in precedenza, uno dei metodi più efficaci per imparare la programmazione degli AVR è guardare degli esempi di codice funzionanti, e capire come funzionano. Nelle prossime pagine troverete moltissimi spunti da utilizzare e personalizzare a vodtro piacimento. ■

DIMMER PER MOTO

Con questo circuito potrete controllare la luminosità di una o due lampade a 6/12V. Un gadget ideale per la vostra moto

Lo schema di questo dimmer a microcontrollore è mostrato in **figura 1.1**. L'uscita del generatore PWM del micro è collegato la gate del MOSFET il quale trasmette l'onda quadra del PWM alla lampada. Con i componenti riportati è possibile pilotare cacichi fino a 20W, per carichi maggiori sarà necessario utilizzare un MOSFET in grado di sopportare maggiore corrente.
Il cuore del sistema è un ATmega8 con quarzo esterno, ma è possibile anche utilizzare l'oscillatore interno apportando le dovute modifiche al codice.
Il circuito prevede un regolatore di tensione a 5V per l'alimentazione per cui può essere connesso direttamente ai 12V della batteria. Utilizzando un micro in package DIP, potrete saldare i componenti direttamente sui pin del microcontrollore vista la semplicità del circuito.
Una buona alternative è comunque l'uso di una millefori o, meglio ancora, di un circuito stampato realizzato appositamente. Si noti nello schema che il MOSFET è TTL compatibile per cui può essere connesso direttamente al micro senza alcun componente di interfaccia. La bassa resistenza in conduzione di questo MOSFET ci garantisce inoltre una bassissima potenza dissipata quando la lampada è accesa. Nello schema è riportata solo una lampada, ma se volete controllare due lampada indipendenti non dovrete far altro che replicare la sezione MODFE/lampada/switch per il secondo carico.
Il software prevede già il controllo per due lampade e, per essere consistenti con lo schema proposto, la parte software di controllo del secondo carico è commentata. Per la compilazione del software è necessario utilizzare le librerie AVRX che trovate comunque nel download.
Il funzionamento operativo è piuttosto semplice. Premendo e rilasciando il pulsante si aumenta la luminosità secondo quattro gradi predefiniti. Premendo e mantenendo premuto il pulsante, si diminuisce la luminosità. Se la luminosità è già al minimo, premendo e mantenendo premuto il pulsante, si provoca lo spegnimento della lampada. All'accensione la lampada si trova sempre al massimo della luminosità. Intervenendo nel codice del programma potrete cambiare sia i gradi di luminosità che il numero di step preimpostati. ■

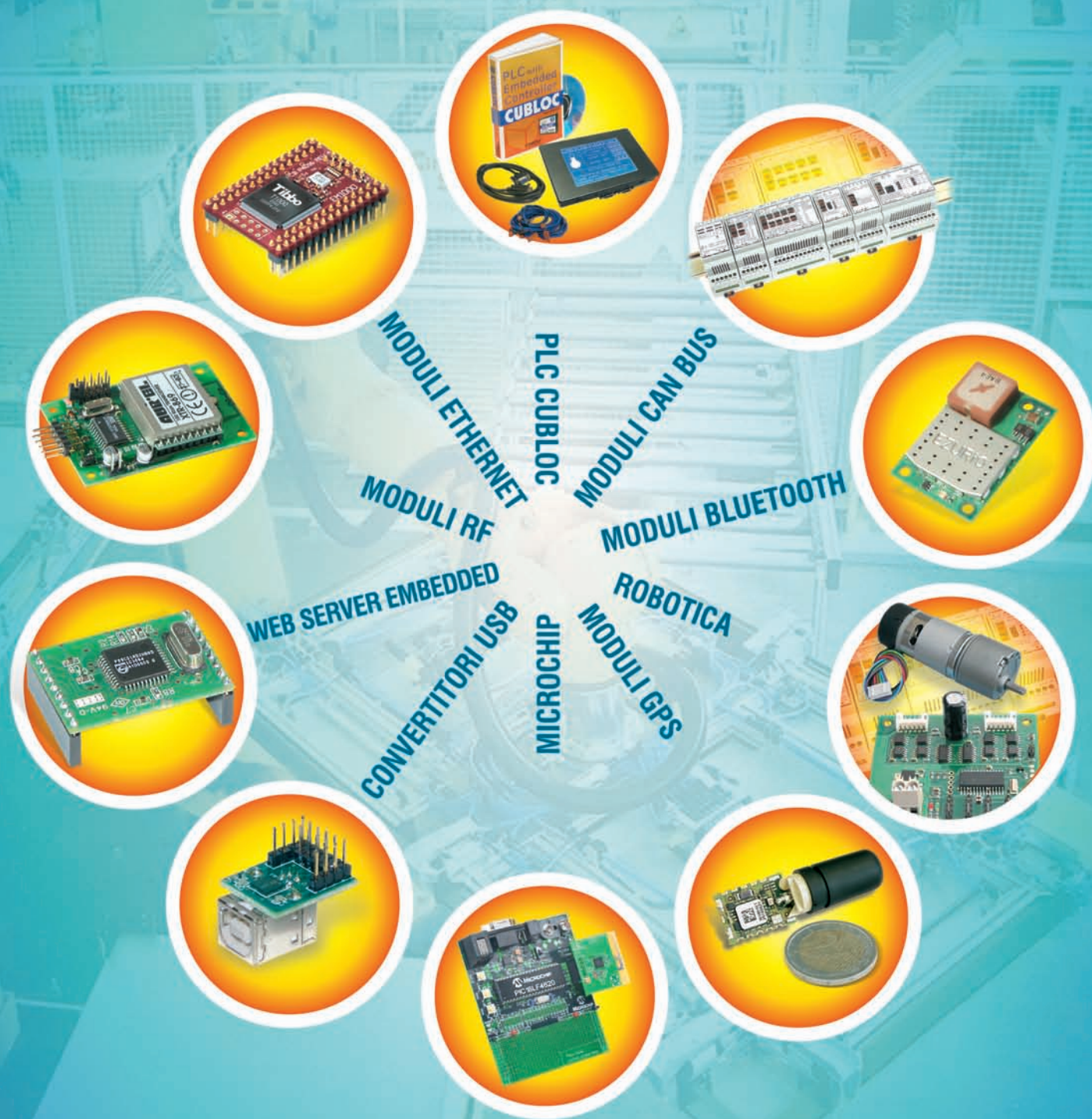



Figura 1.1: schema elettrico del dimmer per moto.

LISTA COMPONENTI			
IC1	= MEGA8-P	Q3	= quarzo 8MHz
C1	= 100n	R1	= 10k
C2	= 100n	R2	= 150
C3	= 22p	S1	= Pulsante N.A.
C4	= 22p	T1	= P30NE03

Microcontrollore
ATmega8

Compilatore
WinAVR

GESTIONE MEMORIE COMPACT FLASH

Ecco un semplice schema e le relative routine software per gestire una memoria CF con un AT90S2313

Microcontrollore
AT90S2313

Compilatore
AVR Studio 4

Con il circuito di **figura 2.1** potrete leggere e scrivere dati su una memoria Compact Flash da un PC utilizzando un collegamento RS232 con la memoria stessa.
Il firmware dell'AT90S2313 gestisce u semplice protocollo di comunicazione con il PC consentendo l'accesso alla CF (il circuito è stato testato con una memoria da 32Mb) per poter leggere e scrive dati. E' possibile utilizzare memorie Compact Flash di qualsiasi dimensione tuttavia si sconsiglia l'uso di memorie di oltre 32Mb in quanto i tempi di trasferimento dati (trasferimento che avviene attraverso la porta seriale) diverrebbero troppo elevati.
La memoria non viene gestita mediante una FAT ma viene letta e scritta settore per settore.
Tra i file disponibili per il download anche i sorgenti del programma di gestione del circuito da parte del PC.Figura 2.1: schema elettrico per l'interfacciamento della memoria Compact Flash.

Figura 2.2: una memoria Compact Flash.

Compact Flash

Le schede di memoria di tipo CompactFlash hanno un formato di 42,8 X 36,4mm con uno spessore che cambia a seconda del tipo; esistono infatti due tipi di questo modello:

- CompactFlash Tipo I con uno spessore di 3,3 mm
- CompactFlash Tipo II con uno spessore di 5 mm

La capacità delle schede CompactFlash varia attualmente da 16 MB a 64 GB a seconda del modello, anche se le attuali specifiche permettono di raggiungere teoricamente i 137 GB.
Con lo standard CF+, revisione 2.0, la velocità di trasferimento è stata aumentata a 16 MB/s (sulla scheda la velocità è indicata usando un fattore di moltiplicazione: ad esempio, "40x" equivale ad una velocità di 6 MB/s). Nel corso del 2005 sono state introdotte delle nuove specifiche tecniche (CF 3.0) che, tra l'altro, supportano una velocità di trasferimento di 66 MB/s.
Lo standard CF 4.0 supporta IDE Ultra DMA 133, che permette di raggiungere una velocità di trasferimento massima di 133 MB/s. Nate inizialmente per le fotocamere digitali, visto il grande successo si sono diffuse anche in altri campi, come ad esempio nell'audio digitale.■

LISTA COMPONENTI

C1	= 100n
C2	= 33p
C3	= 33p
C4	= 470µ
C5	= 100n
C6	= 10µ
C7	= 10µ
C8	= 10µ
C9	= 10µ
IC1	= 74LS573
IC2	= AT90S2313P
IC3	= 78L05
IC4	= MAX232
X1	= Quarzo 4MHz
X2	= Connettore DB9-F
CN1	= Connettore per CF

Figura 2.1: schema eletrico per la gestione delle CF.

OROLOGIO DCF77 CON LCD

Questo progetto impiega un ricevitore per ricevere il segnale orario dalla stazione tedesca DCF77 e lo visualizza su un display LCD alfanumerico HD44780 compatibile

Compilatore
AVR Assembler

DCF77 è il nome della stazione tedesca che trasmette 24 ore su 24 via radio il segnale orario secondo un protocollo ben definito. La diffusione nell'etere del segnale orario ha permesso ai costruttori di radiosveglie di realizzare orologi in grado di rimettersi da soli in base al segnale orario ricevuto appunto via radio. Questo progetto vi permetterà di costruire un orologio con queste caratteristiche, visualizzando il segnale orario DCF77 su un display LCD alfanumerico compatibile con il controller HD44780. Il circuito è quello di **figura 3.1** ed impiega un ATtiny12 con oscillatore RC interno. il display è un LCD alfanumerico 16x2 e viene pilotato impiegando solo tre linee del microcontrollore.
Questo è stato possibile grazie all'impiego di un semplice shift register (il 74CH164) che provvede a convertire e rendere paralleli i dati che arrivano dal microcontrollore.

Orologi radiocontrollati

Gli orologi sincronizzati tramite onde radio inviate da una stazione terrestre raggiungono una accuratezza di circa 1 millisecondo rispetto al tempo standard, dipendente comunque da imprecisioni e variazioni nelle condizioni di propagazione radio. Le stazioni specializzate nel fornire questo servizio hanno le seguenti caratteristiche:

- La frequenza della portante del canale radio usato è correlata alla frequenza standard.
- Un tono speciale identifica l'inizio della codifica dei secondi.
- Permettono di calcolare un intervallo di tempo.
- Diffondono le esatte coordinate geografiche dell'antenna per consentire il calcolo del tempo di propagazione.

In Italia la RAI diffonde il segnale orario sulle reti radiofoniche e televisive, intercalato alle normali trasmissioni. Il segnale è riconoscibile dalla sequenza di sei bip prima dell'annuncio vocale dell'ora esatta. L'ultimo bip indica il momento dello scoccare del minuto.
Il particolare suono che precede i bip è una modulazione digitale a due toni che codifica l'orario standard, fornito dall' Istituto elettrotecnico nazionale. Appositi circuiti elettronici possono decodificare questi dati.
In Germania il segnale orario viene diffuso dalla stazione DCF77, gestita dal Physikalisch-Technische Bundesanstalt, e largamente utilizzata anche in Italia.
La sigla significa: D=Deutschland (Germania), C=segnale ad onde lunghe, F=Francoforte, 77=Frequenza di 77,5 kHz. Le coordinate geografiche del trasmettitore sono: 50°01 N 9°00 E / 50.017, 9.
Il riferimento temporale è mantenuto con un orologio atomico e diffuso da Mainflingen, a circa 25 km di distanza da Francoforte.
Il segnale è ricevibile da una distanza di 2000 km. Sono usati due trasmettitori da 50 kW indipendenti in modo da garantire la continuità del servizio.
Negli USA il segnale, prodotto dagli orologi del NIST (National Institute of Standards and Technology) viene trasmesso in onde lunghe a 60 kHz, potenza 50 kW e con una semplice codifica BCD. L'informazione oraria è anche letta da voci computerizzate e

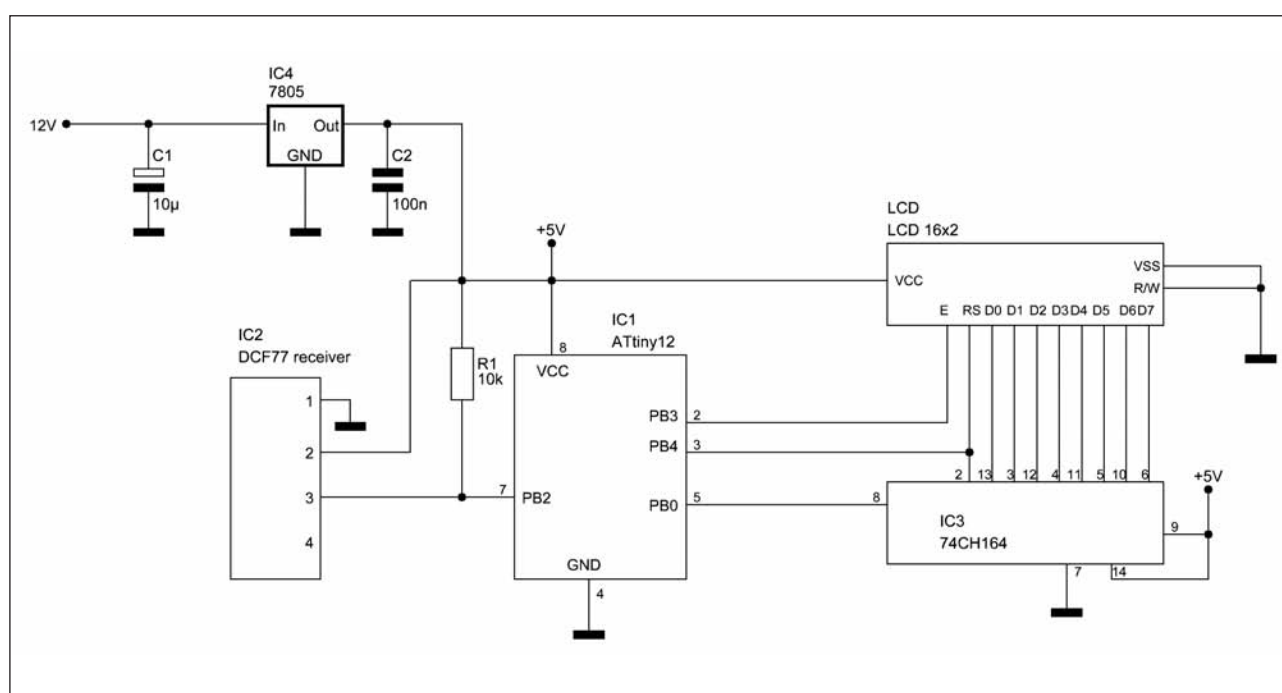



Figura 3.1: schema elettrico dell'orologio DCF77.

LISTA COMPONENTI

C1	= 10µF	IC2	= DCF77 receiver	LCD	= display alfanumerico 16 caratteri su 2 linee
C2	= 100nF	IC3	= 74CH164		
IC1	= ATtiny12	IC4	= LM7805	R1	= 10k

trasmessa su appositi canali radio. In Gran Bretagna la stazione MSF trasmette dal 1° aprile 2007 il campione di tempo e frequenza in onde lunghe alla frequenza di 60 kHz a cura del National Physical Laboratory (NPL) dalla stazione di Anthorn, Cumbria in sostituzione di quello già trasmesso dalla stazione nei pressi di Rugby. Il trasmettitore è gestito da VT Communications per conto del NPL La BBC trasmette dal 1924 sulle sue reti un segnale derivato dall'ora standard di Greenwich.
In alcuni orologi l'ora è ricevuta dal Global Positioning System (GPS), che offre un valore più preciso rispetto alle trasmissioni terrestri. Il sistema GPS combina il tempo fornito da diversi orologi atomici installati a bordo dei satelliti del sistema, e una rete di stazioni terrestri determina e corregge gli errori. Poiché il tempo è ricavato contemporaneamente da diverse sorgenti, l'orologio può automaticamente compensare i ritardi di propagazione ed altri problemi, arrivando ad una accuratezza inferiore al microsecondo in condizioni ideali.
In modo simile al GPS sono utilizzabili il sistema russo GLONASS e il sistema europeo Galileo entrerato in servizio nel 2008. ■

FAN CONTROLLER

Il modo più semplice per variare la velocità di rotazione di una ventola in funzione della temperatura

Il circuito proposto in **figura 4.1** permette di controllare la velocità di rotazione di una ventola a 12V in funzione della temperatura.
La temperatura viene rilevata mediante un NTC e più è alta la temperatura più veloce è la rotazione della ventola. Quest'ultima è un semplice modello a 12V che viene controllato dal microcontrollore con la tecnica PWM attraverso il transistore 2N2219. Il diodo D1 impedisce il danneggiamento della ventola durante le commutazioni ON/OFF del transistor.
E' possibile utilizzare altri modelli di ventola avendo cura di utilizzare per T1 un transistor in grado di supportare la corrente della ventola. L'NTC è inserito nella rete di partizione resistiva insieme a R3.
La tensione su R3 è dunque una partizione dei 5V di alimentazione dipendente dal valore di temperatura a cui è sottoposto l'NTC.
Tale tensione va in ingresso al convertitore AD del micro che provvede a modulare il segnale PWM in funzione di tale valore. ■

Figura 4.1: schema elettrico del FAN controller.

LISTA COMPONENTI

C1	= 10µ 10V	NTC1	= NTC da 10k
C2	= 10µ 25V	R2	= 3,3K
D1	= 1N4148	R3	= 5K
D2	= 1N4148	R4	= 10k
D3	= 1N4148	T1	= 2N2219
IC1	= TINY15LP	VR1	= LM 7805

Microcontrollore
ATtiny15

Compilatore
AVR Studio 4

GAS DETECTOR

Un rilevatore di gas ad allarme acustico

Nel settore della Domotica (l'automazione all'interno della casa), le applicazioni più richieste sono quelle dedicate all'antintrusione ed ai sensori ambientali. Questi dispositivi integrati con i sistemi di sicurezza, controllano l'abitabilità dell'ambiente e le relative situazioni critiche.
Tra i sensori ambientali il più richiesto è certamente il rilevatore di gas; questo dispositivo interviene quando la percentuale di gas presente nell'ambiente supera un livello definito critico per la salute e per uso della stanza. Per essere avvisati da una sirena se nel vostro appartamento vi sono perdite di gas, potete realizzare il circuito rappresentato in **figura 5.1**.

Lo schema di questo rilevatore di gas, si basa su un sensore MQ-2 prodotto da Hanwei Electronics, azienda specializzata i rilevatori e sensori di fumo e gas. L'MQ-2 è in grado di rilevare la presenza di diversi tipi di gas, tra questi vi sono: Butano, Propano, Metano, Alcohol, Idrogeno e fumo. Il cuore del circuito è un microcontrollore ATmega8 che si occupa della gestione completa del dispositivo: la lettura del segnale del sensore, la visualizzazione delle informazioni sul display, l'accensione della sirena. Il sensore di gas è collegato al pin 23 (programmato come ingresso del convertitore A/D), l'A/D converter legge il livello di tensione (proporzionale al tipo e quantità di gas presente). Mediante i tasti set-up e set-down collegati ai pin 9 e 10 è possibile regolare la soglia di tolleranza del circuito. Se il livello del gas rilevato supera la soglia impostata, viene portata a livello 1 l'uscita del pin 18 che, polarizzando il transistor T2, provvede ad azionare la sirena. Contemporaneamente viene portata a livello alto anche l'uscita relativa al pin 19 che, attraverso il transistor T3 va ad eccitare il relè Rel1. Questo relè può essere utilizzato per azionare un allarme visivo, una allarme acustico supplementare o un qualsiasi dispositivo di intervento come ad esempio un combinatore telefonico o un a ventola. Ai pin 15, 16 e 17 sono collegati a 3 led che indicano il funzionamento normale, lo stato di allarme e l'eccitazione del relè (indicato nello schema come led FAN). L'ampiezza degli passi (step) di impostazione delle soglie e la logica dell'attivazione delle segnalazioni è regolabile modificando alcuni parametri nel firmware del microcontrollore. Nel **listato 1** è visibile la gestione del display LCD e la definizione dei messaggi: `"GAS VALUE : "`, `ALARM SET : "`, `"* SENSOR ERROR *"`, `"** NO  SENSOR **"`. ■

LISTATO 1

```
#include <mega8.h>
// Alphanumeric LCD Module functions
#asm
.equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#define set_up    PINB.6
#define set_down  PINB.7
#define fan       PORTB.5
#define buzzer    PORTB.4
#define fan_led   PORTB.3
#define alarm     PORTB.2
#define normal    PORTB.1
#define backlight PORTB.0
unsigned int gas_data;
eeprom unsigned int gas_set=500;
const unsigned char text_g[]="GAS VALUE : ";
const unsigned char text_a[]="ALARM SET : ";
flash unsigned char text_s[]="* SENSOR ERROR *";
flash unsigned char text_o[]="** NO  SENSOR **";
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | 0x40;
// Delay needed for the stabilization of the ADC
input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;//0100 0000
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
...............continua
```

Microcontrollore
ATmega8

Compilatore
Codevision

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	AC OR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω±5%	Room Tem
P_H	Heating consumption	less than 800mw	

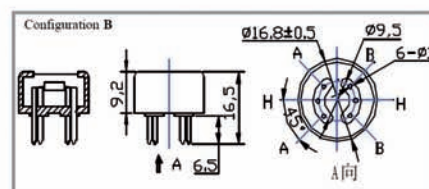
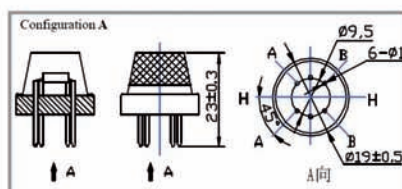
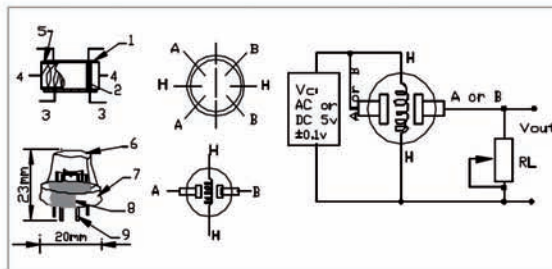
B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-20C-50C	
Tas	Storage Tem	-20C-70C	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Remarks
Rs	Sensing Resistance	3KΩ-30KΩ (1000ppm iso-butane)	Detecting concentration scope: 200ppm-5000ppm LPG and propane 300ppm-5000ppm butane 5000ppm-20000ppm methane 300ppm-5000ppm H_2 100ppm-2000ppm Alcohol
α (3000/1000) isobutane	Concentration Slope rate	≤0.6	
Standard Detecting Condition	Temp: 20C±2C Vc:5V±0.1 Humidity: 65%±5% Vh: 5V±0.1		
Preheat time	Over 24 hour		

D. Structure and configuration, basic measuring circuit

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

E. Sensitivity characteristic curve

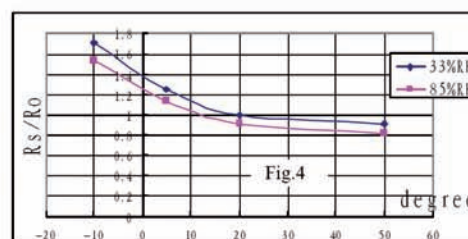
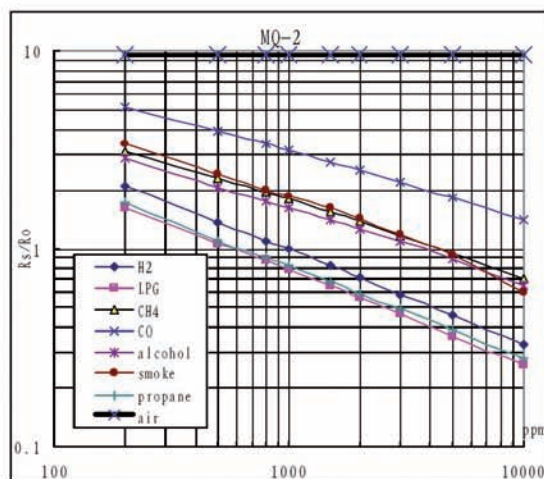
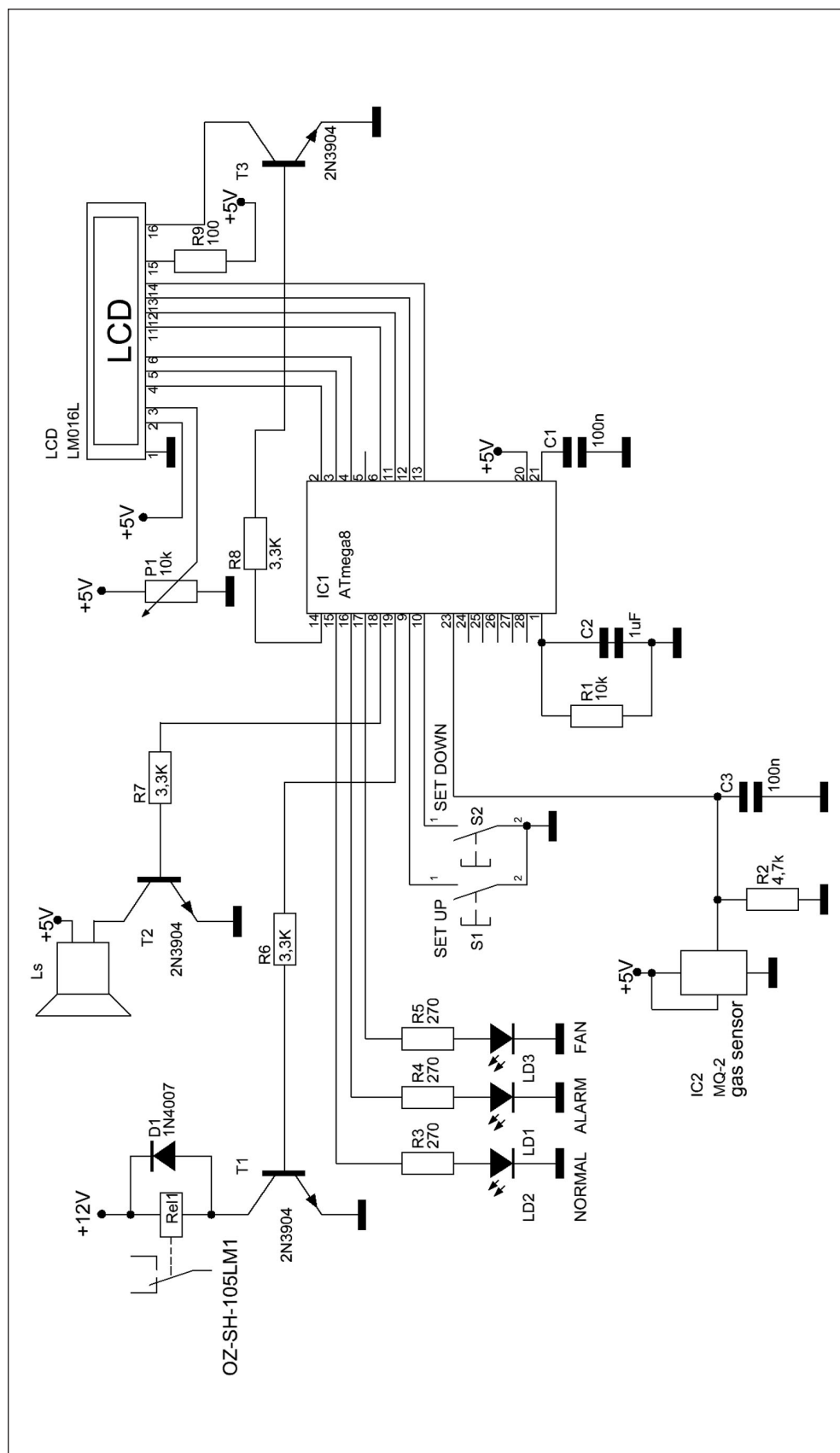



Figure 5.2: caratteristiche del sensore di gas MQ-2.

Figura 5,1: schema elettrico del GAS detector.

LISTA COMPONENTI

C1 = 100n
C2 = 1uF
C3 = 100n

D1 = 1N4007

IC1 = ATmega8
IC2 = MQ-2

LCD = LM016L (Display LCD alfanumerico 16x2)

LD1 = LED
LD2 = LED
LD3 = LED

Ls = Buzzer

P1 = 10k (trimmer)

R1 = 10k
R2 = 4,7k
R3 = 270
R4 = 270
R5 = 270
R6 = 3,3K
R7 = 3,3K
R8 = 3,3K
R9 = 100

Rel1 = relè 1 scambio OZ-SH-105LM1

S1 = Pulsante n.a.
S2 = Pulsante n.a.

T1 = 2N3904
T2 = 2N3904
T3 = 2N3904

GPS DATALOGGER

Un circuito in grado di memorizzare le coordinate geografiche ricevute da un GPS per poi renderle disponibili su un PC attraverso una connessione RS232

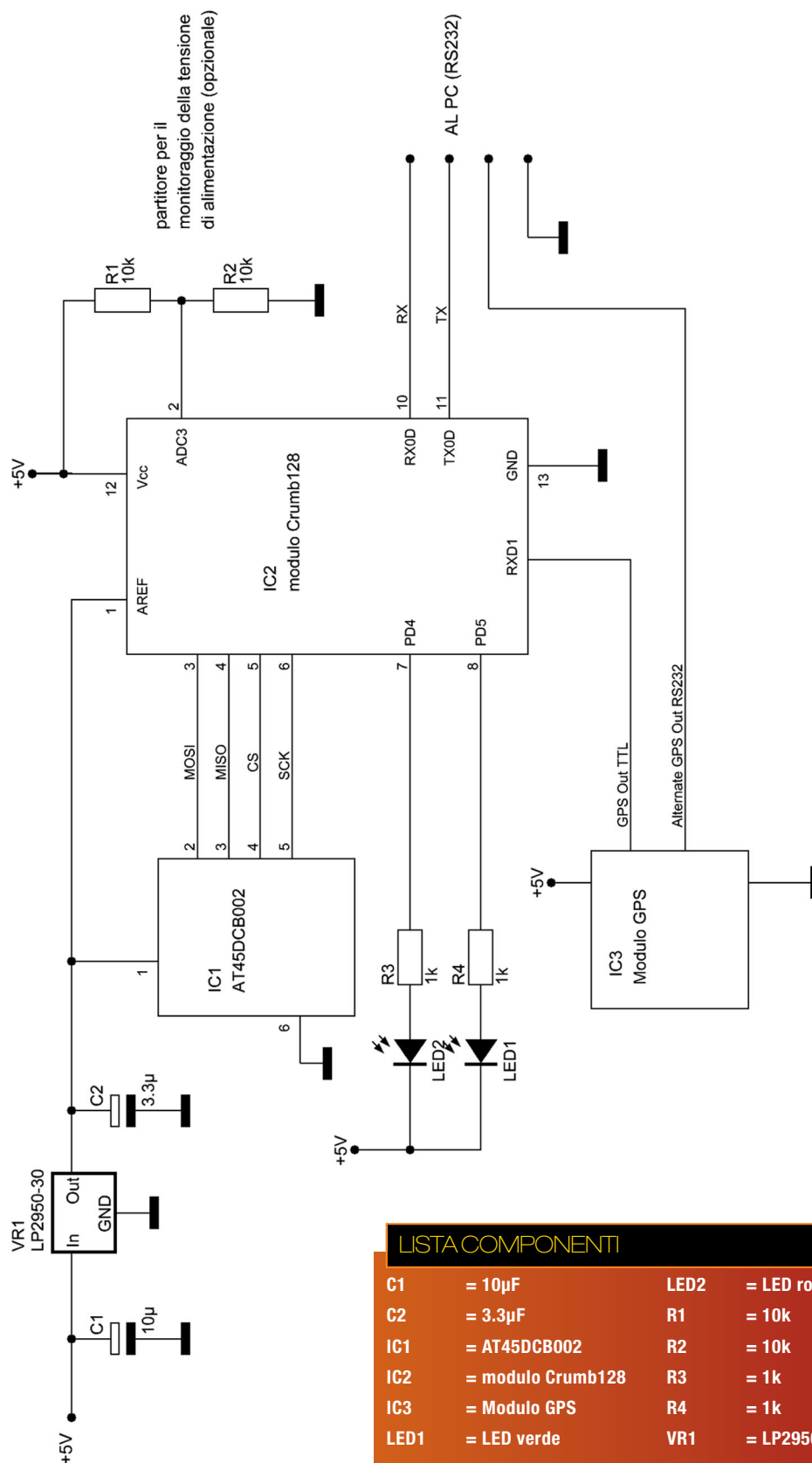
Compilatore
WinAVR

Utilizzando questo circuito è possibile ricevere ogni secondo le coordinate geografiche da un ricevitore GPS e memorizzare su una memoria flash esterna in modo che possano essere trasferite su un PC mediante una connessione seriale RS232. Per avere un ordine di grandezza di quanta memoria flash sia richiesta basti sapere che una memory card da 2MB possono essere registrate ben oltre 60 ore di informazioni provenienti dal GPS.
Il progetto utilizza un ATmega128 montato su una scheda di sviluppo nota come Crumb128. Questa piccola scheda di 40x30mm fornisce sia l'interfaccia RS232 verso il PC sia l'interfaccia seriale TTL da utilizzare per la connessione del GPS ed è alimentata a 5V. Come ricevitore GPS può essere impiegato qualsiasi modello. Per il prototipo è stato utilizzato il Locsense LS-40CM che è dotato di antenna integrata, è alimentato a 5V ed ha l'interfaccia seriale TTL da cui fuoriescono i dati in formato NMEA.
La connessione tra GPS e AVR è ad un solo filo in quanto i dati transitano in modo unidirezionale dal GPS verso il micro.
La memoria per lo storage dei dati è una AT45DCB002 che viene connessa al micro mediante interfaccia SPI. Ovviamente con opportune modifiche si può utilizzare anche una MMC o una SD visto che anch'esse utilizzano una connessione SPI per comunicare con il micro.
l'interfaccia utente è ridotta al minimo: solo due LED forniscono lo stato del sistema: il LED verde lampeggia quando viene ricevuta una stringa NMEA valida, mentre il LED rosso lampeggia ogni volta che viene rilevato un errore.
Ciascuna stringa NMEA ha la seguente struttura:
$<*msg type*><*msg data*><CR><LF>
Il campo $<*msg type*> identifica il tipo di pacchetto dati. Esistono 4 tipi di stringhe NMEA: GSA, GSV, RMC e GGA. Il datalogger processa solo questi ultimi due tipi di stringhe da cui estrae le coordinate. Quando viene ricevuta una stringa NMEA il micro ne controlla il checksum e, se non ci sono errori, ne viene estratto il contenuto e memorizzato in flash. La memoria flash deve necessariamente essere scritta in blocchi la cui dimensione dipende dal tipo di flash. Poiché i dati estratti da una singola stringa NMEA sono di dimensioni molto inferiori al singolo blocco, è necessario che questi vengano bufferizzati dal micro prima di essere scritti in memoria.
Con la memoria utilizzata in questo progetto i dati vengono bufferizzati e scritti in memoria ogni 60 secondi. Oltre alle coordinate viene memorizzato anche un codice di controllo CRC che permetterà di verificare l'integrità dei dati quando questi verranno trasferiti al PC.
Per l'upload dei dati occorre connettere la scheda al PC ed inviare i caratteri #c sulla seriale che porteranno la scheda in modalità comandi.

I satelliti GPS

La costellazione GPS è costituita da 31satelliti attivi. I satelliti supplementari migliorano la precisione del sistema permettendo misurazioni ridondanti. Al crescere del numero di satelliti, la costellazione è stata modificata secondo uno schema non uniforme che si è dimostrato maggiormente affidabile in caso di guasti contemporanei a più satelliti. Il sistema di navigazione si articola nelle seguenti componenti:

- un complesso di 24 satelliti, divisi in gruppi di quattro su ognuno dei sei piani orbitali (distanti 60° fra loro e inclinati di 55° sul piano equatoriale)

Figura 6.1: schema elettrico del ricevitore GPS

LISTA COMPONENTI

C1	**= 10µF**	**LED2**	**= LED rosso**
C2	**= 3.3µF**	**R1**	**= 10k**
IC1	**= AT45DCB002**	**R2**	**= 10k**
IC2	**= modulo Crumb128**	**R3**	**= 1k**
IC3	**= Modulo GPS**	**R4**	**= 1k**
LED1	**= LED verde**	**VR1**	**= LP2950-30**

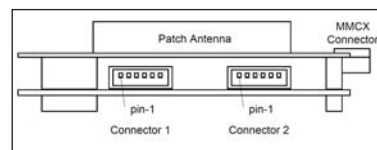
Ricevitore GPS LS-40CM

Caratteristiche
- ricezione a 12 canali paralleli
- supporto per SBAS (WAAS, EGNOS)
- possibilità di connettere direttamente sul modulo una antenna attiva
- Elevata sensibilità:
  - Sensibilità acquisizione: 137dBm
  - Sensibilità tracking:145dBm
- Avvio veloce:
  - < 10 secondi in hot start
  - < 45 secondi in cold start
  - Accuratezza: 5m

Pinout - Connettore 1

Pin	Segnale	Descrizione
1	Serial Data Out 1	Uscita seriale asincrona a livello LVTTL, per l'uscita dei messaggi NMEA
2	Serial Data In 1	Ingresso seriale asincrono a livello LVTTL per l'invio dei comandi di configurazione
3	Serial Data Out 2	Uscita seriale asincrona a livello RS-232, per l'uscita dei messaggi NMEA
4	Serial Data In 2	Ingresso seriale asincrono a livello LVTTL per l'invio dei comandi di configurazione
5	Power	1 3.8V ~ 12.0V DC
6	GND	Massa dell'alimentazione e dei segnali

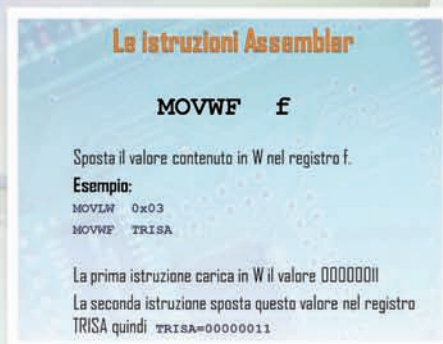
Pinout - Connettore 2

Pin	Segnale	Descrizione
1	GND	Massa dell'alimentazione e dei segnali
2	GND	Massa dell'alimentazione e dei segnali
3	NC	Non connesso
4	Serial Data In 1	Ingresso seriale asincrono a livello LVTTL per l'invio dei comandi di configurazione
5	Serial Data Out 1	Uscita seriale asincrona a livello LVTTL, per l'uscita dei messaggi NMEA
6	Power	2 3.3V DC

- 2 cicli al giorno
- una rete di stazioni di tracciamento (tracking station)
- un centro di calcolo (computing station)
- due stazioni di soccorrimento (injection stations)
- un ricevitore GPS.

I 31 satelliti della costellazione GPS sono disposti su 6 piani orbitali inclinati di 55° rispetto al piano equatoriale (quindi non coprono le zone polari) a forma di ellissi a bassa eccentricità. Ogni piano orbitale ha 3 o 4 satelliti, e i piani sono disposti in modo tale che ogni utilizzatore sulla terra possa ricevere i segnali di almeno 5 satelliti. La loro quota è di 20 200 km e compiono due orbite complete in un giorno siderale. Ciascun satellite emette sulle frequenze di 1,2 e 1,5 GHz derivate da un unico oscillatore ad alta stabilità. Lo scopo della doppia frequenza è quello di eliminare l'errore dovuto alla rifrazione atmosferica. Su queste frequenze portanti, modulate in fase, vengono emessi i messaggi di effemeride, ciascuno della durate di due minuti; essi iniziano e terminano ai minuti pari interi del GMT. Questi messaggi di effemeride contengono il segnale orario e i parametri orbitali del satellite. In tal modo il ricevitore GPS, mentre effettua il conteggio doppler, riceve i parametri dell'orbita da cui deriva la posizione del satellite: viene così a disporre di tutti gli elementi necessari a definire nello spazio la superficie di posizione.

In orbita vi sono 24 satelliti per la trasmissione di dati GPS, più 3 di scorta. Da questo si evince che da un punto del globo terrestre il ricevitore riesce a vedere solo la metà di essi, quindi 12.

Ma non li vedrà mai tutti e 12 per via della loro inclinazione rispetto all'equatore. In più il ricevitore GPS stesso fa una discriminazione dei satelliti: preferisce quelli più vicino possibile alla perpendicolare per questione di ricezione del timing in quanto il dato da quelli con più inclinazione arriverebbe con maggiore ritardo. Ogni satellite è dotato di 4 oscillatori ad altissima precisione, di cui 2 al cesio e 2 al rubidio; ha dei razzi per effettuare le correzioni di orbita.

Ha due pannelli solari di area pari a 7,25 m² per la produzione di energia. Ha infine batterie di emergenza per garantire l'apporto energetico nei periodi in cui il sole è eclissato. Pesa circa 845 kg ed ha una vita di progetto di 7,5 anni. ■

TERMOMETRO MULTICANALE

Un circuito in grado di monitorare la temperatura in punti diversi impiegando fino a 16 sensori one-wire DS1820

Questo circuito è in grado di gestire fino a 16 sensori one-wire Dallas DS1820 per monitorare la temperatura in altrettanti punti dell'ambiente. I dati vengono visualizzati su un display LCD alfanumerico e possono essere inviati ad un PC esterno mediante connessione RS232 per le elaborazioni relative.
La frequenza di lettura dei sensori può essere impostata da 5 a 9999 secondi e ciascun dato letto viene registrato associandogli il timestamp ricavato dalla sezione RTC (che comunque può anche non essere montata).
All'accensione sul display viene visualizzato il numero di sensori rilevati sul bus 1-wire. Dopo alcuni secondi se viene rilevata la sezione RTC connessa al bus viene visualizzato il relativo messaggio.
Nel caso di presenza di RTC i valori di temperatura verranno trasmessi in coppia con data e ora di rilevamento. Se l'RTC non è presente come timestamp viene preso il tempo trascorso dall'ini-

Figura 7.1: pinout del sensore Dallas DS18S20.

Microcontrollore
ATmega168

Compilatore
AVR Studio 4

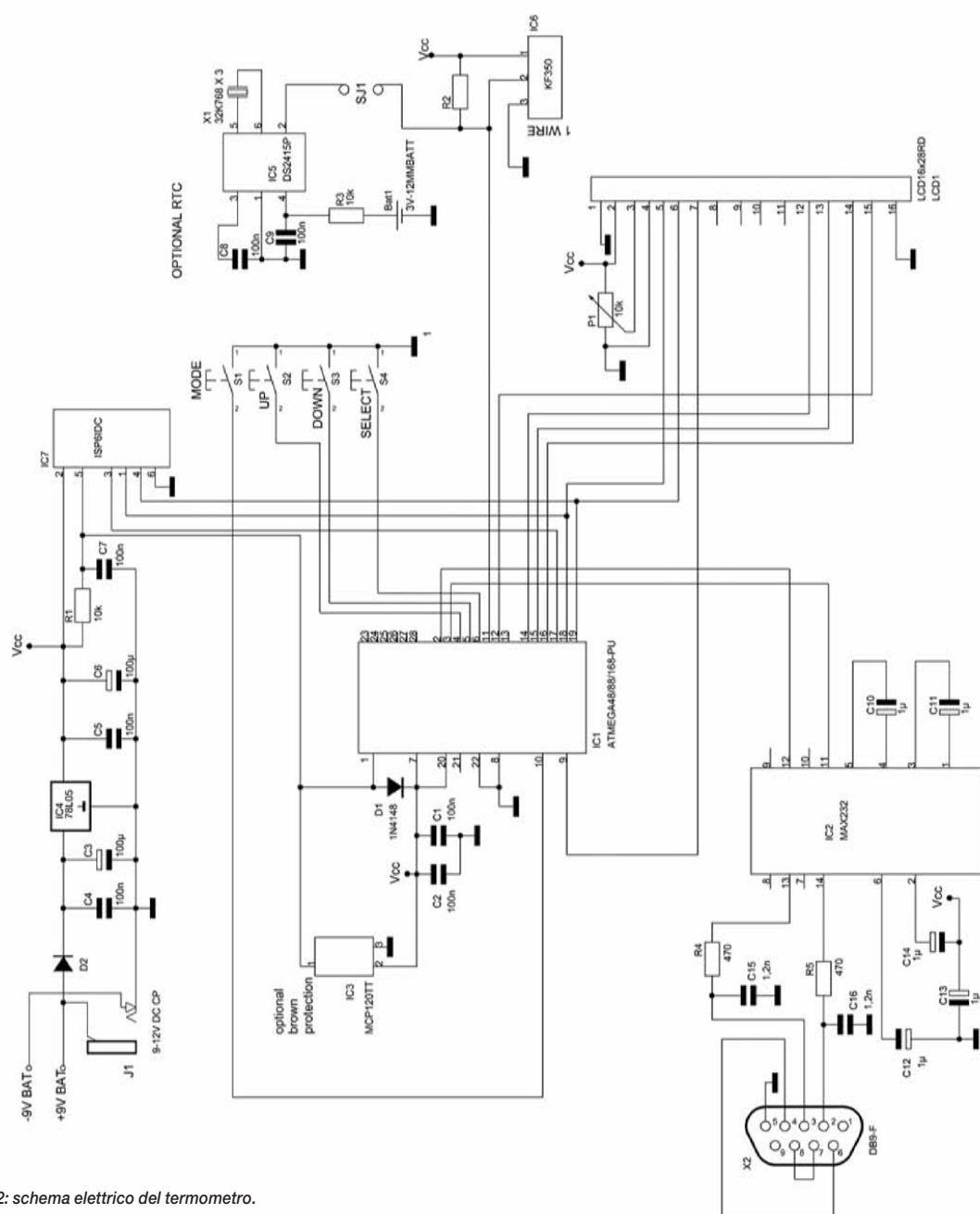



Figura 7.2: schema elettrico del termometro.

LISTA COMPONENTI

Bat1	= Batteria 3V	C11	= 1µ	IC3	= MCP120TT	R2	= 1k
C1	= 100n	C12	= 1µ	IC4	= 78L05	R3	= 10k
C2	= 100n	C13	= 1µ	IC5	= DS2415P	R4	= 470
C3	= 100µ	C14	= 1µ	IC6	= KF350	R5	= 470
C4	= 100n	C15	= 1,2n	IC7	= ISP6IDC	S1	= pulsante n.a.
C5	= 100n	C16	= 1,2n	J1	= 9-12V DC CP (Jack 5mm)	S2	= pulsante n.a.
C6	= 100µ	D1	= 1N4148			S3	= pulsante n.a.
C7	= 100n	D2	= 1N4002	LCD1	= LCD16x28RD (display alfanumerico)	S4	= pulsante n.a.
C8	= 100n	IC1	= ATMEGA48/88/168 PU				
C9	= 100n			P1	= 10k	X1	= 32K768 X 3 (quarzo)
C10	= 1µ	IC2	= MAX232	R1	= 10k	X2	= DB9-F (Connettore)

PIN DESCRIPTION

PIN		NAME	FUNCTION
TO-92	SO		
1	5	GND	Ground
2	4	DQ	Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the *Powering the DS18S20* section.)
3	3	V_{DD}	Optional V_{DD}. V_{DD} must be grounded for operation in parasite power mode.
—	1, 2, 6, 7, 8	N.C.	No Connection

Figura 7.3: i segnali ai pin del DS18S20.

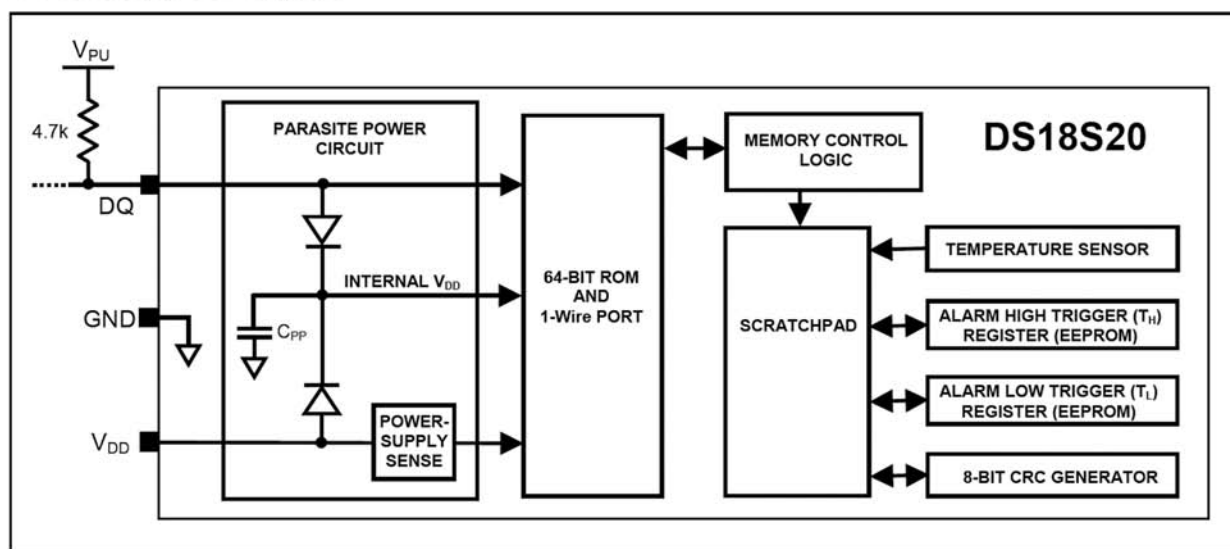



Figura 7.4: schema a blocchi del sensore DS18S20.

zio del rilevamento. Per aggiungere o rimuovere i sensori è necessario prima spegnere il sistema in modo cha alla riaccensione venga aggiornato automaticamente il numero di sensori rilevati.
Mediante il tasto MODE è possibile impostare l'ordine di visualizzazione delle letture dai vari sensori. Per cambiare l'ordine si può agire sui pulsanti UP, DOWN e SELECT. Salvare i cambiamenti col tasto MODE. Gli intervalli di lettura possono essere rapidamente impostati mantenendo premuti i tasti UP o DOWN. Confermare l'intervallo con il tasto MODE. Mediante il tasto MODE si accede inoltre alla schermata che consente l'aggiustamento dell'intensità della retroilluminazione del display. I valori possibili vanno da 0 (retroilluminazione disattivata) a 255 (massima illuminazione) impostabili mediante i tasti UP e DOWN. Il valore scelto viene salvato nella memoria non volatile e verrà ricaricato ad ogni avvio. Se nel circuito è presente l'RTC, premendo MODE si può accedere alla schermata di regolazione della data e dell'ora.
La modalità logging viene attivata premendo UP, DOWN o SELECT durante la schermata di startup. Vengono visualizzate le schermate relative ad ogni singolo sensore secondo l'ordine impostato. Per ciascuna schermata viene visualizzato il valore precedentemente letto, il valore corrente e l'identificativo del sensore. I dati vengono trasmessi anche sulla porta seriale a 19200,8,N,1 e possono quindi essere ricevuti usando un qualsiasi software di emulazione di terminale (ad esempio il classico HyperTerminal di Windows). ■

IL SET DELLE ISTRUZIONI

Per scrivere e modificare programmi assembly è necessario conoscere le istruzioni che il micro mette a disposizione. Ecco allora il set delle istruzioni dei micro AVR con la spiegazione dell'operazione, i registri coinvolti ed il numero di cicli macchina necessari all'esecuzione dell'istruzione stessa.

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
		Arithmetic and Logic Instructions				
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1	
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1	
ADIW	Rd, K	Add Immediate to Word	Rd ← Rd + 1:Rd + K	Z,C,N,V,S	2 [1]	
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1	
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1	
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1	
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1	
SBIW	Rd, K	Subtract Immediate from Word	Rd + 1:Rd ← Rd + 1:Rd - K	Z,C,N,V,S	2 [1]	
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1	
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1	
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1	
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1	
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1	
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V,S	1	
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1	
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1	
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1	
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1	
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1	
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1	
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1	
SER	Rd	Set Register	Rd ← $FF	None	1	
MUL	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2 [1]	
MULS	Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2 [1]	
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2 [1]	
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2 [1]	
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2 [1]	
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2 [1]	
DES	K	Data Encryption	if (H = 0) then R15:R0 ← Encrypt(R15:R0, K) else if (H = 1) then R15:R0 ← Decrypt(R15:R0, K)			1/2
		Branch Instructions				
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2	
IJMP		Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2 [1]	
EIJMP		Extended Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	2 [1]	
JMP	k	Jump	PC ← k	None	3 [1]	

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3 / 4[4]	2 / 3[4]
ICALL		Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	3 / 4[1][4]	2 / 3[1][4]
EICALL		Extended Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	4 [1][4]	3 [1][4]
CALL	k	call Subroutine	PC ← k	None	4 / 5[1][4]	3 / 4[4]
RET		Subroutine Return	PC ← STACK	None	4 / 5[4]	
RETI		Interrupt Return	PC ← STACK	I	4 / 5[4]	
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3	
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1	
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1	
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1	
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3	
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1 / 2 / 3	
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3	2 / 3 / 4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC ← PC + 2 or 3	None	1 / 2 / 3	2 / 3 / 4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1 / 2	
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1 / 2	
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2	
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2	
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2	
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2	
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2	
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2	
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2	
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2	
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2	
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2	
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2	
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2	
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2	
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2	
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2	
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2	
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2	
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2	
Data Transfer Instructions						
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1	
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1 [1]	
LDI	Rd, K	Load Immediate	Rd ← K	None	1	
LDS	Rd, k	Load Direct from data space	Rd ← (k)	None	2[1][4]	2[4][5]
LD	Rd, X	Load Indirect	Rd ← (X)	None	2[2][4]	1[4][5]

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X) X ← X + 1	None	$2^{(2)(4)}$	$1^{(4)(5)}$
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, ← X - 1 Rd ← (X) ← (X)	None	$2^{(2)(4)}$	$2^{(4)(5)}$
LD	Rd, Y	Load Indirect	Rd ← (Y) ← (Y)	None	$2^{(2)(4)}$	$1^{(4)(5)}$
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y) Y ← Y + 1	None	$2^{(2)(4)}$	$1^{(4)(5)}$
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1 Rd ← (Y)	None	$2^{(2)(4)}$	$2^{(4)(5)}$
LDD	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	$2^{(1)(4)}$	$2^{(4)(5)}$
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	$2^{(2)(4)}$	$1^{(4)(5)}$
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	$2^{(2)(4)}$	$1^{(4)(5)}$
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	$2^{(2)(4)}$	$2^{(4)(5)}$
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	$2^{(1)(4)}$	$2^{(4)(5)}$
STS	k, Rr	Store Direct to Data Space	(k) ← Rd	None	$2^{(1)(4)}$	$2^{(4)}$
ST	X, Rr	Store Indirect	(X) ← Rr	None	$2^{(2)(4)}$	$1^{(4)}$
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	$2^{(2)(4)}$	$1^{(4)}$
ST	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	$2^{(2)(4)}$	$2^{(4)}$
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	$2^{(2)(4)}$	$1^{(4)}$
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	$2^{(2)(4)}$	$1^{(4)}$
ST	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	$2^{(2)(4)}$	$2^{(4)}$
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	$2^{(1)(4)}$	$2^{(4)}$
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	$2^{(2)(4)}$	$1^{(4)}$
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr Z ← Z + 1	None	$2^{(2)(4)}$	$1^{(4)}$
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1	None	$2^{(2)(4)}$	$2^{(4)}$
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	$2^{(1)(4)}$	$2^{(4)}$
LPM		Load Program Memory	R0 ← (Z)	None	$3^{(3)}$	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	$3^{(3)}$	3
LPM	Rd, Z+	Load Program Memory and Post-Increment	Rd ← (Z), Z ← Z + 1	None	$3^{(3)}$	3
ELPM		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	$3^{(1)}$	
ELPM	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	$3^{(1)}$	
ELPM	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd ← (RAMPZ:Z), Z ← Z + 1	None	$3^{(1)}$	
SPM		Store Program Memory	(RAMPZ:Z) ← R1:R0	None	$-^{(1)}$	-
SPM	Z+	Store Program Memory and Post-Increment by 2	(RAMPZ:Z) ← R1:R0, Z ← Z + 2	None		-
IN	Rd, A	In From I/O Location	Rd ← I/O(A)	None	1	
OUT	A, Rr	Out To I/O Location	I/O(A) ← Rr	None	1	
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	$2^{(1)}$	$1^{(4)}$
POP	Rd	Pop Register from Stack	Rd ← STACK	None	$2^{(1)}$	$2^{(4)}$

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
		Bit and Bit-test Instructions				
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0, C ← Rd(7)	Z,C,N,V,H	1	
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0, C ← Rd(0)	Z,C,N,V	1	
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V,H	1	
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1	
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1	
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1	
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1	
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1	
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	2	1
CBI	A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	2	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1	
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1	
SEC		Set Carry	C ← 1	C	1	
CLC		Clear Carry	C ← 0	C	1	
SEN		Set Negative Flag	N ← 1	N	1	
CLN		Clear Negative Flag	N ← 0	N	1	
SEZ		Set Zero Flag	Z ← 1	Z	1	
CLZ		Clear Zero Flag	Z ← 0	Z	1	
SEI		Global Interrupt Enable	I ← 1	I	1	
CLI		Global Interrupt Disable	I ← 0	I	1	
SES		Set Signed Test Flag	S ← 1	S	1	
CLS		Clear Signed Test Flag	S ← 0	S	1	
SEV		Set Two's Complement Overflow	V ← 1	V	1	
CLV		Clear Two's Complement Overflow	V ← 0	V	1	
SET		Set T in SREG	T ← 1	T	1	
CLT		Clear T in SREG	T ← 0	T	1	
SEH		Set Half Carry Flag in SREG	H ← 1	H	1	
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1	
		MCU Control Instructions				
BREAK		Break	(See specific descr. for BREAK)	None	1(1)	
NOP		No Operation		None	1	
SLEEP		Sleep	(see specific descr. for Sleep)	None	1	
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1	

NOTE:

(1) Questa istruzione non è disponibile in tutti i dispositivi. Fare riferimento al datasheet del dispositivo specifico.

(2) Non tutte le varianti di questa istruzione sono disponibili in tutti i dispositivi. Fare riferimento al datasheet del dispositivo specifico.

(3) Non tutte le varianti dell'istruzione LPM sono disponibili in tutti i dispositivi. Fare riferimento al datasheet del dispositivo specifico.

(4) I cicli di clock per l'accesso alla memoria dati fanno riferimento alla memoria dati interna del dispositivo e non sono validi in caso di utilizzo di una memoria esterna.

(5) In caso di accesso alla SRAM interna si deve considerare un ciclo in più.

GENERATORE DI BARRE VIDEO

Con un AT90S2313 e pochi altri componenti aggiuntivi potrete realizzare questo semplicissimo generatore di barre video in standard PAL

Il circuito è basato su un AT90S2313 operante a 17.734475MHz ed i segnali digitali generati dal micro vengono convertiti in analogico da una semplice rete R2R a 10 resistori. La frequenza di lavoro è oltre il 77% di quella supportata dal micro. Per poche ore di funzionamento il micro non ne risentirà, ma se volete usare il circuito per periodi prolungati è consigliato passare ad un microcontrollore operante a frequenza più elevata. Collegando il circuito alla presa scart del televisore verranno visualizzate a video cinque barre colorate verticali. Se decidete di modificare il file sorgente tenete conto del numero di cicli istruzione in quanto aggiungendo o rimuovendo istruzioni andrete a modificare i tempi di esecuzione del programma che potrebbe in questo modo generare una immagine poco stabile.

Lo standard PAL

Il Phase Alternating Line o PAL è un metodo di codifica del colore utilizzato nella televisione, usato da gran parte del mondo. Fanno eccezione il continente americano, alcune nazioni dell'est asiatico, parte del Medio Oriente, dell'Europa orientale e la Francia. Il PAL fu sviluppato in Germania da Walter Bruch, che lavorava alla Telefunken, e fu usato per la prima volta nel 1967. Alle stesse conclusioni, nello stesso periodo dei tedeschi arrivò anche una squadra di tecnici RAI nei laboratori di Torino, ma l'invenzione rimase in un cassetto e fu quindi attribuita giustamente alla Telefunken che la pubblicò per prima. Il termine "PAL" è spesso usato informalmente per riferirsi al

LISTA COMPONENTI

C1	= 22p	R4	= 1k
C2	= 22p	R6	= 1k
CN1	= Connettore RCA	R7	= 1k
Cr1	= 17,73 MHz (quarzo)	R8	= 1k
IC1	= AT90S2313	R9	= 500
R1	= 1k	R10	= 500
R2	= 1k	R11	= 500
R3	= 500	Tutte le resistenze sono da 1/4W.	

Microcontrollore
AT90S2313

Compilatore
AVR Studio 4

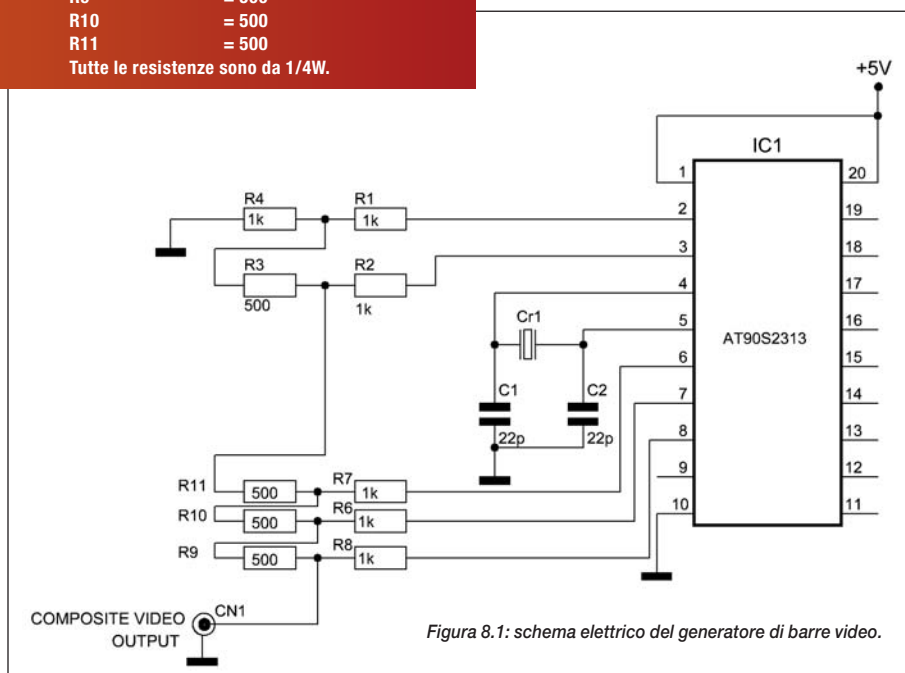



Figura 8.1: schema elettrico del generatore di barre video.

formato video digitale a 625 linee/50 Hz (576i, comune nei paesi europei), così come il termine "NTSC" indica spesso il formato a 525 linee/60 Hz (480i), usato in Nord America e in Giappone. Questo uso del termine è scorretto ma comunque di largo impiego, tanto che spesso i supporti DVD sono etichettati come "PAL" o "NTSC" anche se dal punto di vista tecnico sul DVD non c'è un segnale a colori composito codificato PAL. È importante sottolineare il fatto che il termine PAL non indica numeri di linee o frequenze di scansione, ma indica esclusivamente il sistema usato per la codifica dei colori della sottoportante. Per contrasto, lo standard NTSC definisce il numero di linee e dei semiquadri. Nell'uso comune, quindi, si considera come PAL un formato video interlacciato a 625 linee orizzontali, dove 50 semiquadri formano 25 fotogrammi al secondo, con una frequenza di 50 Hz, che andrebbe più comunemente chiamato CCIR 601. Per realizzare un'immagine conforme allo standard PAL con un programma di elaborazione d'immagini, le dimensioni devono essere di 720 pixel in orizzontale e di 576 pixel in verticale, con una risoluzione di 72 pixel/pollice, considerando che esistono dei margini oltre i quali l'immagine non viene visualizzata sullo schermo televisivo e dei margini al di fuori dei quali è meglio non posizionare testi. Per una corretta visualizzazione è fondamentale che all'immagine sia applicato un filtro di interlacciamento. Il formato è un'evoluzione dell'NTSC e possiede una portante di luminanza (luma) di 3,5 MHz di banda passante su cui viene sommata una sottoportante di crominanza (croma) di 4,43361875 MHz. La portante luma (Y) è formata da:
0,30 R + 0,59 G + 0,11 B.
La sottoportante croma viene modulata vettorialmente con le sole componenti R-Y e B-Y e da esse in fase di demodulazione viene ricostruita la componente G-Y. Lo standard che definisce il sistema PAL fu pubblicato dall'Unione Internazionale delle Telecomunicazioni nel 1998 con il titolo "Recommendation ITU-R BT.470-6, Conventional Television Systems". ■

AVR PINOUT

In queste pagine sono riportati i pinout dei microcontrollori AVR utilizzati in questo volume.

ATtiny15

ATmega32

ATmega128

ATmega168

ATtiny12

ATtiny13
8-PDIP/SOIC
(PCINT5/RESET/ADC0/dW) PB5 1
(PCINT3/CLKI/ADC3) PB3 2
(PCINT4/ADC2) PB4 3
GND 4
8 VCC
7 PB2 (SCK/ADC1/T0/PCINT2)
6 PB1 (MISO/AIN1/OC0B/INT0/PCINT1)
5 PB0 (MOSI/AIN0/OC0A/PCINT0)
ATtiny13
20-QFN/MLF
DNC DNC DNC DNC DNC
20 19 18 17 16
(PCINT5/RESET/ADC0/dW) PB5 1
(PCINT3/CLKI/ADC3) PB3 2
DNC 3
DNC 4
(PCINT4/ADC2) PB4 5
15 VCC
14 PB2 (SCK/ADC1/T0/PCINT2)
13 DNC
12 PB1 (MISO/AIN1/OC0B/INT0/PCINT1)
11 PB0 (MOSI/AIN0/OC0A/PCINT0)
ATtiny13
6 7 8 9 10
DNC DNC GND DNC DNC
NOTE: Bottom pad should be soldered to ground.
DNC: Do Not Connect
10-QFN/MLF
(PCINT5/RESET/ADC0/dW) PB5 1
(PCINT3/CLKI/ADC3) PB3 2
DNC 3
(PCINT4/ADC2) PB4 4
GND 5
10 VCC
9 PB2 (SCK/ADC1/T0/PCINT2)
8 DNC
7 PB1 (MISO/AIN1/OC0B/INT0/PCINT1)
6 PB0 (MOSI/AIN0/OC0A/PCINT0)
ATtiny13
NOTE: Bottom pad should be soldered to ground.
DNC: Do Not Connect

AT90S2313
PDIP/SOIC
RESET 1
(RXD) PD0 2
(TXD) PD1 3
XTAL2 4
XTAL1 5
(INT0) PD2 6
(INT1) PD3 7
(T0) PD4 8
(T1) PD5 9
GND 10
20 VCC
19 PB7 (SCK)
18 PB6 (MISO)
17 PB5 (MOSI)
16 PB4
15 PB3 (OC1)
14 PB2
13 PB1 (AIN1)
12 PB0 (AIN0)
11 PD6 (ICP)
AT90S2313

ATmega88
PDIP
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)
ATmega88

ATmega8
PDIP
(RESET) PC6 1
(RXD) PD0 2
(TXD) PD1 3
(INT0) PD2 4
(INT1) PD3 5
(XCK/T0) PD4 6
VCC 7
GND 8
(XTAL1/TOSC1) PB6 9
(XTAL2/TOSC2) PB7 10
(T1) PD5 11
(AIN0) PD6 12
(AIN1) PD7 13
(ICP1) PB0 14
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
24 PC1 (ADC1)
23 PC0 (ADC0)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK)
18 PB4 (MISO)
17 PB3 (MOSI/OC2)
16 PB2 (SS/OC1B)
15 PB1 (OC1A)
ATmega8
TQFP Top View
32 PD2 (INT0)
31 PD1 (TXD)
30 PD0 (RXD)
29 PC6 (RESET)
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
(INT1) PD3 1
(XCK/T0) PD4 2
GND 3
VCC 4
GND 5
VCC 6
(XTAL1/TOSC1) PB6 7
(XTAL2/TOSC2) PB7 8
24 PC1 (ADC1)
23 PC0 (ADC0)
22 ADC7
21 GND
20 AREF
19 ADC6
18 AVCC
17 PB5 (SCK)
ATmega8
9 (T1) PD5
10 (AIN0) PD6
11 (AIN1) PD7
12 (ICP1) PB0
13 (OC1A) PB1
14 (SS/OC1B) PB2
15 (MOSI/OC2) PB3
16 (MISO) PB4
MLF Top View
32 PD2 (INT0)
31 PD1 (TXD)
30 PD0 (RXD)
29 PC6 (RESET)
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
(INT1) PD3 1
(XCK/T0) PD4 2
GND 3
VCC 4
GND 5
VCC 6
(XTAL1/TOSC1) PB6 7
(XTAL2/TOSC2) PB7 8
24 PC1 (ADC1)
23 PC0 (ADC0)
22 ADC7
21 GND
20 AREF
19 ADC6
18 AVCC
17 PB5 (SCK)
ATmega8
9 (T1) PD5
10 (AIN0) PD6
11 (AIN1) PD7
12 (ICP1) PB0
13 (OC1A) PB1
14 (SS/OC1B) PB2
15 (MOSI/OC2) PB3
16 (MISO) PB4

Ecco un circuito per realizzare un contagiri elettronico basato su un sensore ad infrarossi

CONTAGIRI

Compilatore
Codevision AVR

Microcontrollore
AT90S2313

Il sensore impiegato in questo circuito emette un raggio di luce ad infrarossi che può essere riflesso o meno dagli oggetti vicini. Il sensore riconosce che un oggetto è nelle vicinanze in quanto riesce a captare il raggio riflesso.
Con questo metodo risulta piuttosto semplice misurare i giri dell'albero del trapano, la velocità di rotazione della ventola del vostro PC e in tutte quelle applicazioni in cui non è possibile installare un sensore sulle parti rotanti. La **figura 9.1** riporta le caratteristiche del sensore prodotto da Fairchild. Il circuito non richiede particolari tarature, sarà sufficiente orientare il fascio ad infrarossi emesso dal sensore verso la parte in movimento di cui si vuol misurare la velocità di rotazione. Il numero di giri rilevato viene visualizzato su 4 cifre realizzate con display a 7 segmenti. I 4 display dono pilotati in multiplex consentendo così l'utilizzo di sole sette linee dati per il pilotaggio di tutte le 4 cifre. Nello schema è inoltre previsto un connettore per la porta seriale a cui è sufficiente aggiungere un transceiver RS232 per avere una porta seriale perfettamente funzionante e utilizzabile per lo scambio dati con un PC. ■

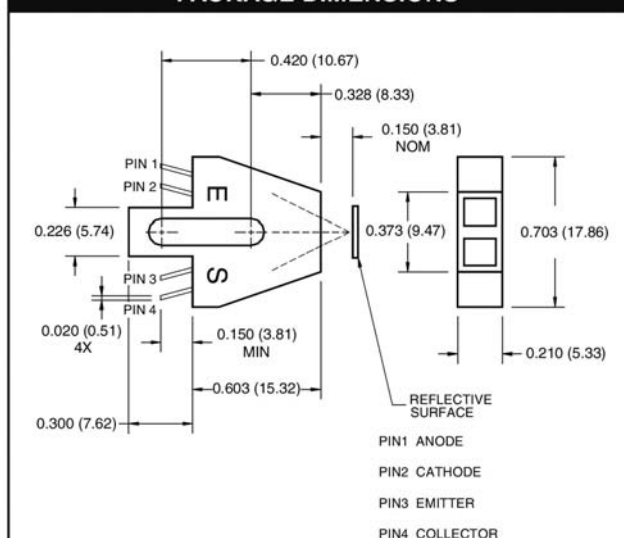
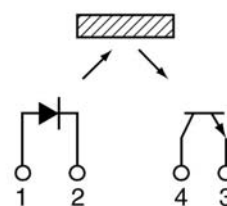
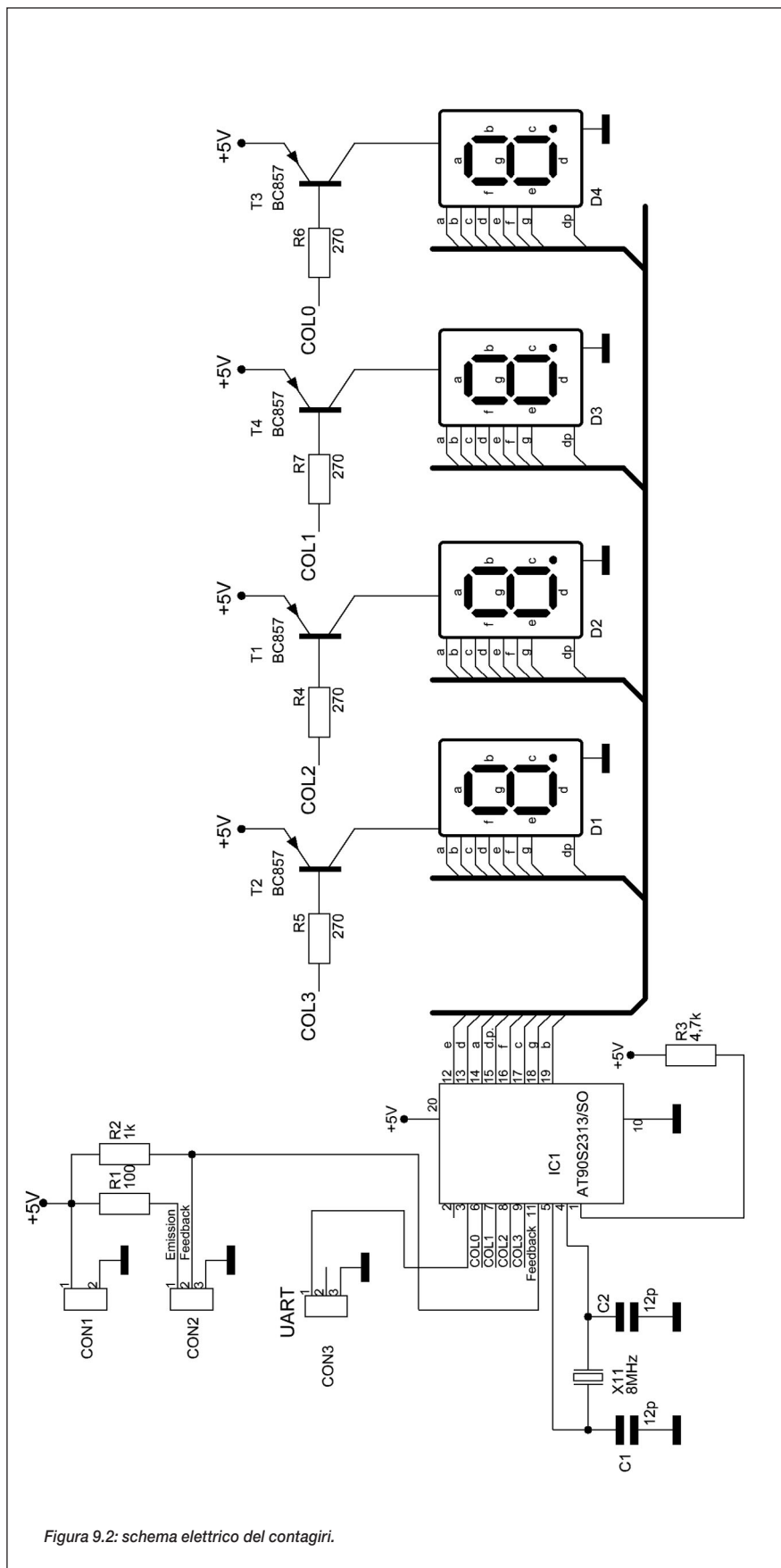



Figura 9.1: caratteristiche del sensore QRB113.

LISTA COMPONENTI

C1	= 12p
C2	= 12p
CON1	= Connettore 2 poli
CON2	= Connettore 2 poli
CON3	= Connettore 2 poli
D1	= display 7 segmenti (catodo comune)
D2	= display 7 segmenti (catodo comune)
D3	= display 7 segmenti (catodo comune)
D4	= display 7 segmenti (catodo comune)
IC1	= AT90S2313/SO
R1	= 100
R2	= 1k
R3	= 4,7k
R4	= 270
R5	= 270
R6	= 270
R7	= 270
T1	= BC857
T2	= BC857
T3	= BC857
T4	= BC857
X11	= 8MHz (quarzo)

Figura 9.2: schema elettrico del contagiri.

SVEGLIA CON LCD GRAFICO

Un orologio realizzato con un ATmega32 ed un display LCD grafico. La particolarità? La gestione intelligente della retroilluminazione del display

Microcontrollore
ATmega32

Compilatore
WinAVR

L'orologio presentato in questo progetto presenta una particolarità introvabile nei modelli commerciali: la gestione della retroilluminazione. E' possibile infatti impostare un intervallo di tempo entro il quale la retroilluminazione è attiva ed un livello di intensità luminosa dell'ambiente al di sopra della quale la retroilluminazione si attiva automaticamente. Potrete ad esempio disattivare la retroilluminazione nelle ore notturne in modo che la luce del display non disturbi il vostro sonno, ma se vi alzate in piena notte ed accendete la luce la retroilluminazione si attiverà automaticamente rendendo visibile l'orario.

Il funzionamento è piuttosto intuitivo: il pulsante ENTER consente di accedere al menu all'interno del quale ci si sposta con i pulsanti UP/DOWN. Una volta scelta la voce di menu vi si accede con il tasto ENTER. Dal menu è possibile impostare, oltre all'orario e la data, anche gli orari di accensione e spegnimento della retroilluminazione nonché il livello di luminosità oltre il quale la stessa retroilluminazione si attiva. I valori impostati per la retroilluminazione vengono salvati nella EEPROM del micro cosicché qualora venisse a mancare la tensione di alimentazione non sarà necessario impostare nuovamente questi valori.

Per rimettere l'ora premere ENTER quindi con il tasto DOWN muoversi di una linea verso il basso e confermare ancora con ENTER. A questo punto le ore lampeggiano e possono essere impostate con i tasti UP/DOWN. Confermando con ENTER lampeggeranno i minuti. Regolate i minuti come già descritto in precedenza e confermate il valore con ENTER. Allo stesso modo si possono impostare il giorno, il mese e l'anno.

Lo schema elettrico del progetto è riportato in **figura 10.1**: il cuore del sistema è un ATmega32 la cui struttura interna è riportata in **figura 10.2**. Potete migliorare il progetto inserendo una batteria di backup che sia in grado di mantenere le impostazioni anche in man-

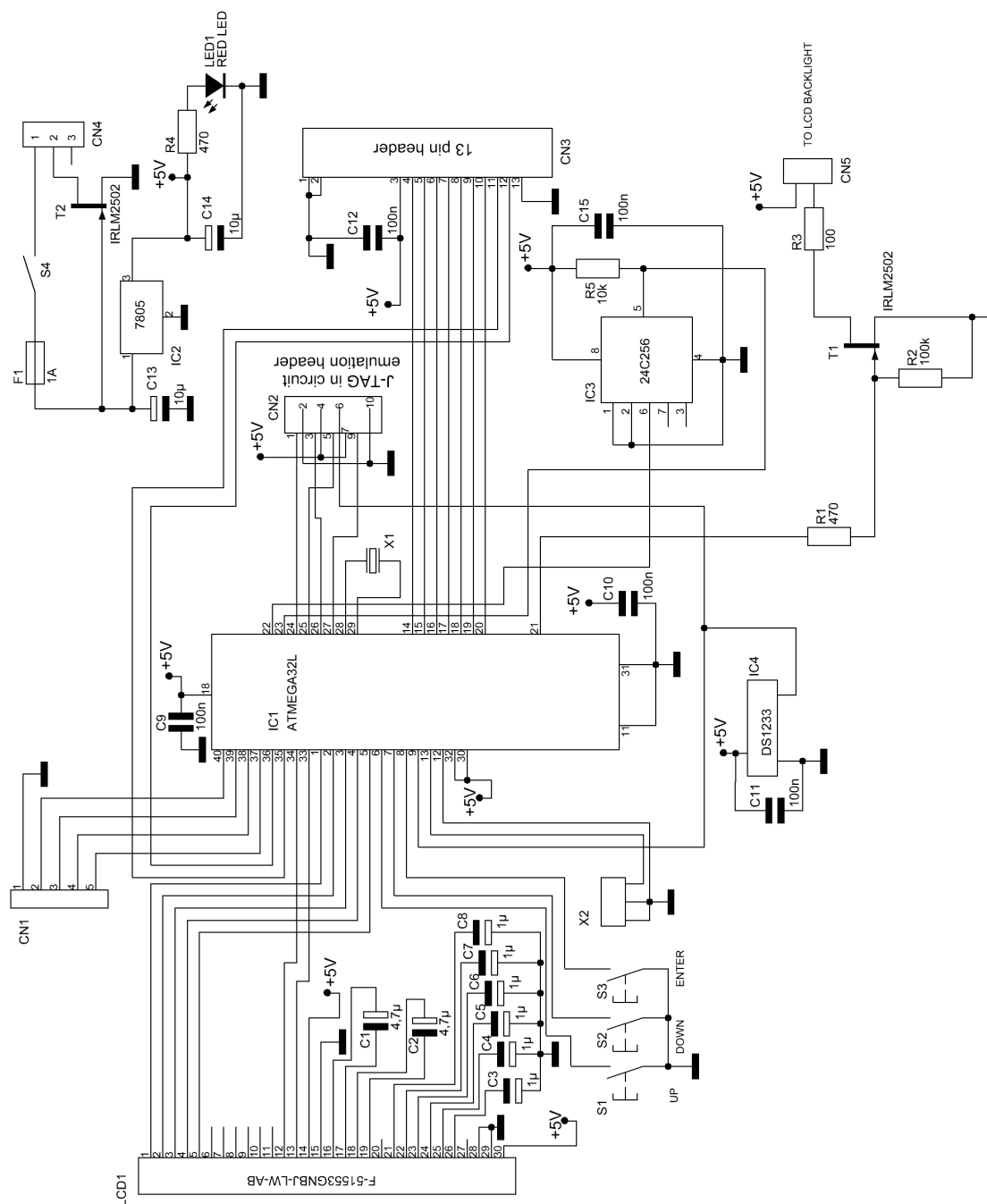



Figura 10.1: schema elettrico della sveglia con LCD grafico.

LISTA COMPONENTI

- C1 = 4,7µ
- C2 = 4,7µ
- C3 = 1µ
- C4 = 1µ
- C5 = 1µ
- C6 = 1µ
- C7 = 1µ
- C8 = 1µ
- C9 = 100n
- C10 = 100n
- C11 = 100n
- C12 = 100n
- C13 = 10µ
- C14 = 10µ
- C15 = 100n
- CN1 = connettore 5 poli per c.s.
- CN2 = connettore ICD-10
- CN3 = Connettore 13 pin (Display grafico 128x64)
- CN4 = connettore 2 poli
- CN5 = connettore 2 poli
- F1 = 1A (fusibile)
- IC1 = ATMEGA32L
- IC2 = 7805
- IC3 = 24C256
- IC4 = DS1233
- LCD1 = F-51553GNBJ-LW-AB
- LED1 = RED LED
- R1 = 470
- R2 = 100k
- R3 = 100
- R4 = 470
- R5 = 10k
- S1 = Pulsante n.a.
- S2 = Pulsante n.a.
- S3 = Pulsante n.a.
- S4 = interruttore
- T1 = IRLM2502
- T2 = IRLM2502
- X1 = quarzo 32 KHz
- X2 = Risuonatore ceramico 4MHz

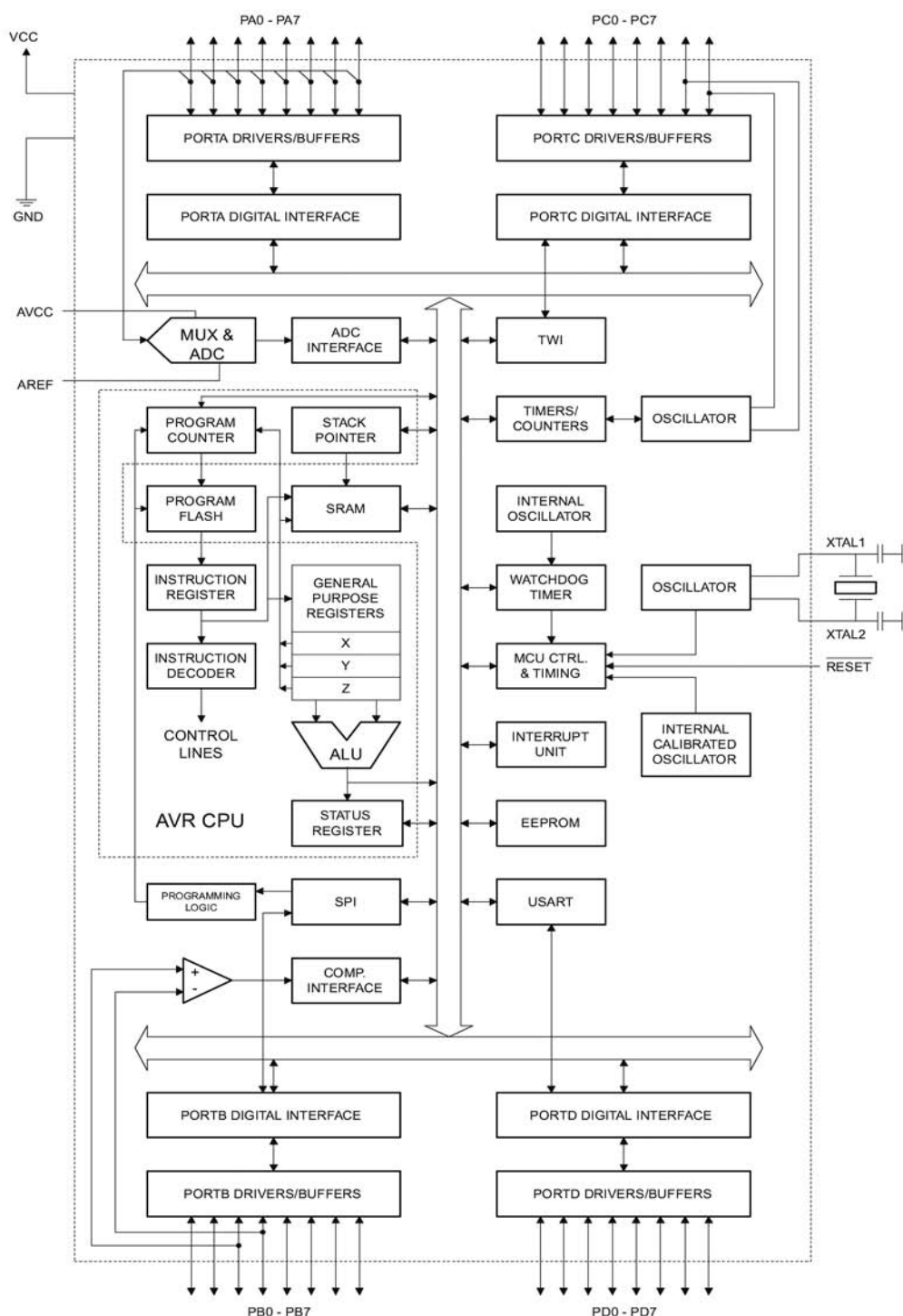



Figura 10.2: la struttura interna dell'ATmega32.

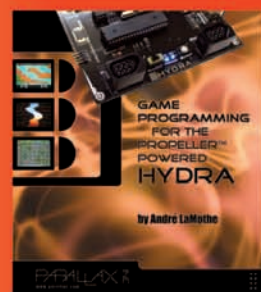
canza della tensione di rete. Per l'uso della batteria è consigliabile ridurre al massimo i tempi in cui la retroilluminazione resta attiva per non limitare la durata della batteria stessa. Altra modifica che può essere fatta è l'utilizzo di una membrana touch screen per l'interazione con l'utente. In questo caso dovranno essere previste sul display i tasti virtuali in corrispondenza delle voci del menu e le routine di gestione del touch screen. Ovviamente l'uso di un display con touch screen permette di evitare l'uso dei pulsanti meccanici esterni.
Per ulteriori informazioni sull'uso delle membrane touch screen si veda il progetto numero 16. ■

Solo 225 EUR!

Progetta giochi, grafica ed applicazioni multimediali con il Propeller™ Powered HYDRA™ Game Console. Per utilizzare questo kit è richiesta solo una piccola esperienza di programmazione BASIC, C o similari. Tutto l'hardware e il software è incluso, insieme ad un bellissimo libro sulla programmazione di giochi con il Propeller nel linguaggio Spin™ e assembly. Inoltre, l'hardware HYDRA è descritto dettagliatamente mediante schemi, descrizioni, e suggerimenti che permettono di utilizzare appieno tutte le risorse, inclusa la sua porta di espansione e la scheda per i giochi. *Questo è quanto troverai all'interno...*

- Software IDE per il Propeller
- Descrizione approfondita del Propeller a livello dei registri
- Programmazione in linguaggio Spin
- Programmazione in Assembly
- Archittura di HYDRA e descrizione dei circuiti
- Grafica 2D e programmazione animata
- Algoritmi dei giochi e pattern software
- Programmazione dei suoni mediante tecniche PCM
- Programmazione dei dispositivi di input: tastiera, mouse e game controller
- Tecniche di ottimizzazione per la programmazione dei giochi
- Modellazione fisica di base per i giochi
- Tecniche di intelligenza artificiale
- Matematica per i giochi: vettori e trasformazioni affini
- Dozzine di demo per sperimentare tutti i topics discussi

(codice 32360; EUR 225,00)

Ordina l'HYDRA Game Development Kit (cod. 32360 EUR 225,00) online su www.elettroshop.com o chiama il numero 02-66504794

PARALLAX
www.parallax.com

I PREZZI INDICATI SONO IVA ESCLUSA

Propeller e Spin sono marchi registrati da Parallax Inc. HYDRA è un marchio registrato da Nurve Networks LCC.

TESTO **SCORREVOLE**

Con questo semplice progetto potrete visualizzare dei testi scorrevoli su una matrice di LED 15x7

Un minipannello a scritte scorrevoli realizzato con 105 LED ed un ATmega8. Ciascuna riga e colonna è pilotata da un transistor che fornisce la corrente necessaria a pilotare tutti i LED connessi. I transistors sono a loro volta pilotati dal microcontrollore che azionando in sequenza ogni singola colonna crea l'effetto di testo scorrevole. La velocità di scorrimento, il senso ed il testo da visualizzare possono essere impostati via software. Inoltre il pannello è modulare per cui si possono aggiungere altre matrici di LED per aumentare la lunghezza del testo visualizzato. ■

LISTA COMPONENTI

IC1 = MEGA8-P

LED1-LED105= LED rosso 5mm

R1 = 33k
R2 = 33k
R3 = 33k
R4 = 33k
R5 = 33k
R6 = 33k
R7 = 33k

T1 –T22 = 2N1711

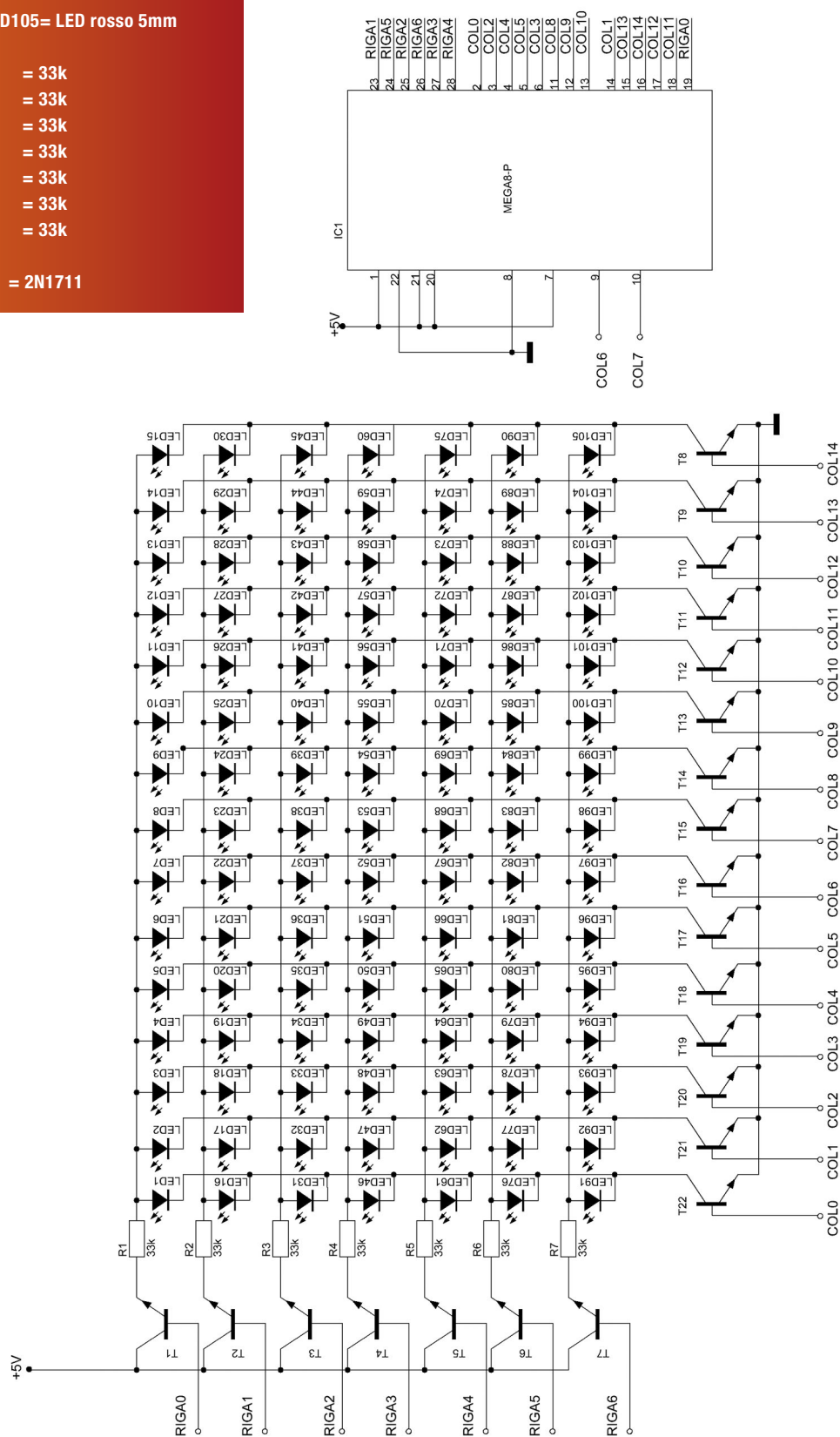



Figura 11.1: schema elettrico del progetto

TOUCH SWITCH

Un interruttore tattile il cui funzionamento è basato sui tempi di carica e scarica di un condensatore autocostruito

Il principio di funzionamento di questo progetto si basa sul tempo di scarica di un condensatore. Il microcontrollore carica la capacità (che è l'elemento sensibile dell'interruttore touch) che poi si scarica attraverso una resistenza. Il tempo di scarica dipende dalla capacità per cui misurando questo tempo attraverso il micro è possibile risalire al valore della capacità. Ovviamente il valore della capacità varia a seconda che si tocchi o meno il sensore. Il sensore è costituito da una basetta in vetronite per circuiti stampati monofaccia di circa 3x3cm su cui è stato praticato un taglio che divide lo strato di rame in due zone a cui sono saldati due terminali. E' consigliabile ricoprire il rame con uno strato di nastro adesivo al fine di evitare interferenze di tipo resistivo.

La **figura 12.1** mostra lo schema elettrico del progetto. In fase di taratura è possibile variare il valore della resistenza di carica/scarica qualora il tempo di carica e scarica siano troppo lunghi o troppo brevi. Un valore maggiore di resistenza allungherà i tempi citati. L'uscita in questo caso pilota un relè ma è anche possibile pilotare altri dispositivi (LED, Buzzer, ecc...) avendo cura di utilizzare la dovuta circuiteria di interfaccia.

Potrete utilizzare questo schema per realizzare un tasto di trasmissione morse per radioamatori senza alcuna parte meccanica in movimento! ■

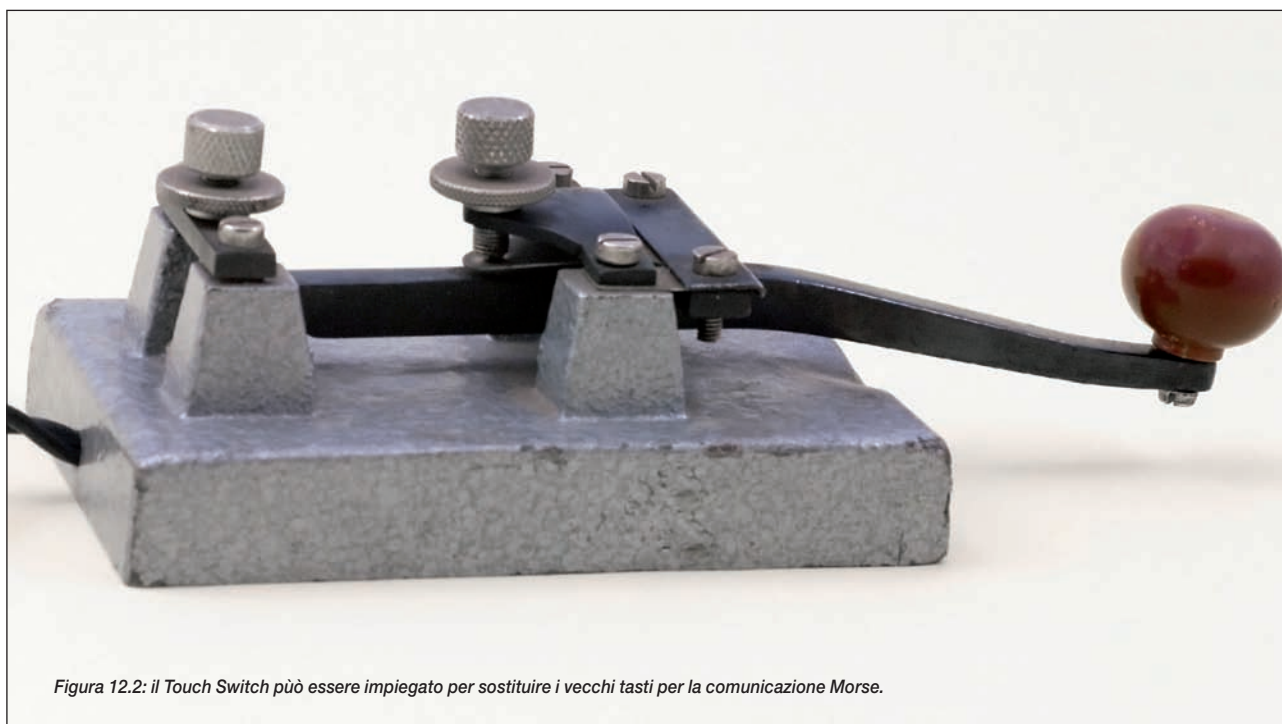



Figura 12.2: il Touch Switch può essere impiegato per sostituire i vecchi tasti per la comunicazione Morse.

LISTA COMPONENTI

D1	= 1N4148	R2	= 10k	S1	= Interruttore
		R3	= 1k	S2	= Sensore touch (vedi testo)
IC1	= ATtiny13	Rel1	= relè 5V a uno scambio		
R1	= 1M			T1	= 2N1711

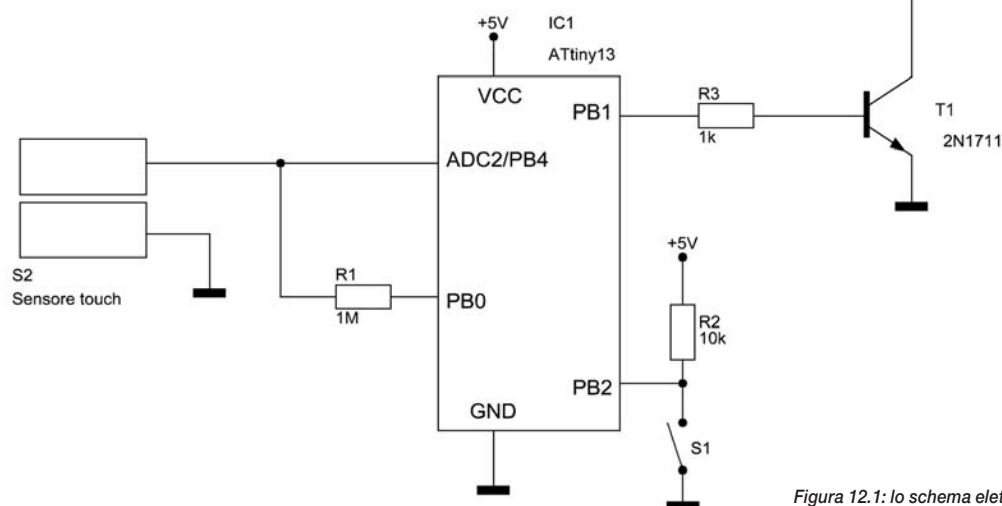



Figura 12.1: lo schema elettrico del progetto.

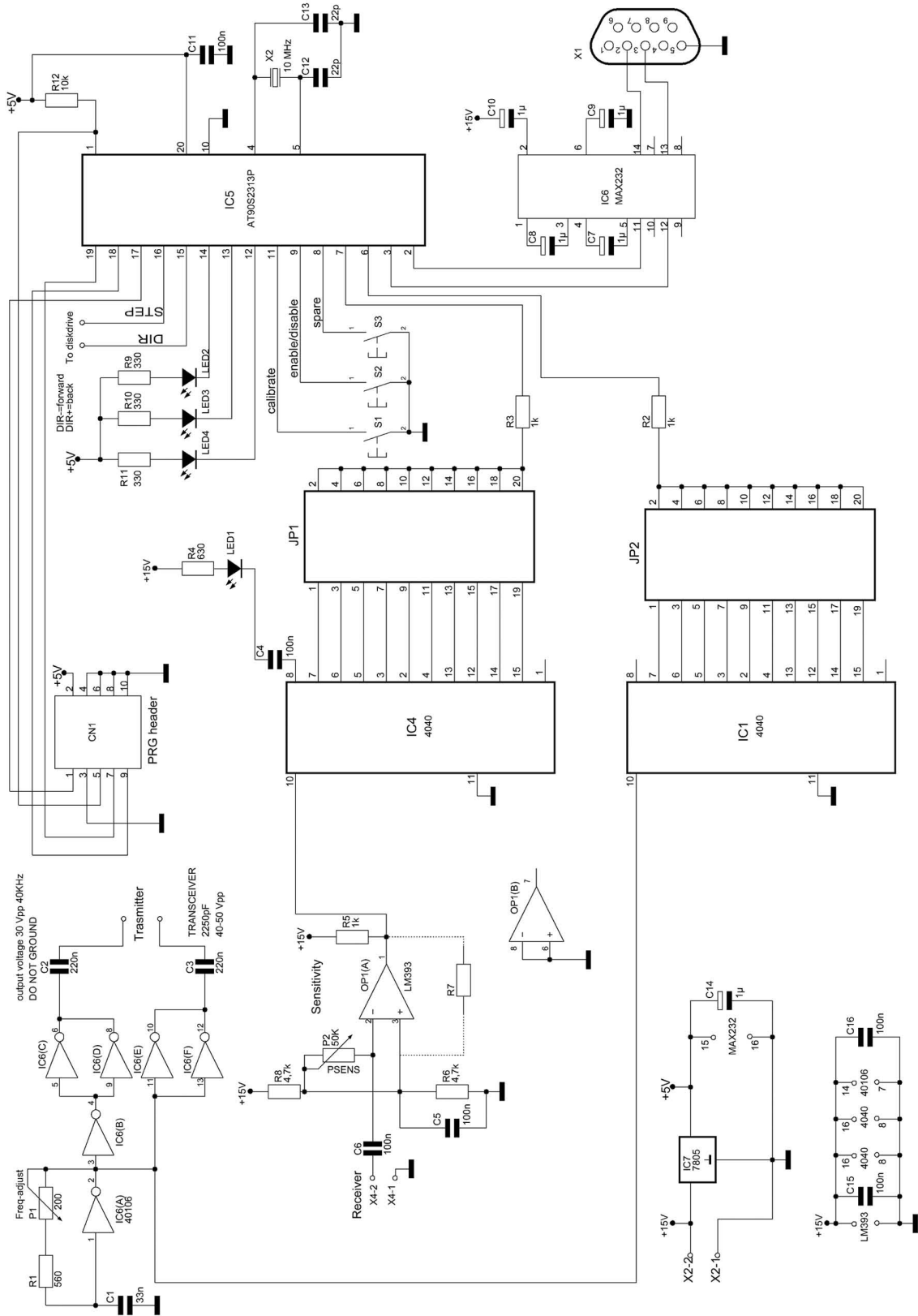
METRO AD ULTRASUONI

Un interessante progetto per la misurazione delle distanze sfruttando la fase di un segnale ad ultrasuoni

La tecnica utilizzata in questo progetto per la misurazione delle distanze è detta “phase shifting” e consiste nell’inviare un segnale costante (una sinusoide a 40KHz) e misurare la differenza di fase tra il segnale trasmesso e quello ricevuto. La differenza di fase è infatti proporzionale alla distanza percorsa dal segnale (distanza calcolata come andata più ritorno). ■

LISTA COMPONENTI

C1	= 33n
C2	= 220n
C3	= 220n
C4	= 100n
C5	= 100n
C6	= 100n
C7	= 1µ
C8	= 1µ
C9	= 1µ
C10	= 1µ
C11	= 100n
C12	= 22p
C13	= 22p
C14	= 1µ
C15	= 100n
C16	= 100n
CN1	= Connettore 10 poli
IC1, IC4	= CD4040
IC5	= AT90S2313P
IC6	= MAX232
IC7	= 7805
IC6	= CD40106
JP1-JP2	= Jumpers 10 vie
LED1	= LED 5mm
LED2	= LED 5mm
LED3	= LED 5mm
LED4	= LED 5mm
OP1	= LM393
P1	= 200 (trimmer)
P2	= 50K (trimmer)
R1	= 560
R2	= 1k
R3	= 1k
R4	= 630
R5	= 1k
R6	= 4,7k
R7	= 1k
R8	= 4,7k
R9	= 330
R10	= 330
R11	= 330
R12	= 10k
S1	= pulsante n.a.
S2	= pulsante n.a.
S3	= pulsante n.a.
X1 (femmina)	= Connettore DB9
X2	= Quarzo 10 MHz

Figura 13.1: lo schema elettrico del progetto.

MONITOR PER BATTERIE

Questo circuito effettua la scarica di una batteria monitorando tutti i parametri in modo da ricostruire la curva caratteristica e la resistenza interna

Un circuito per la diagnosi accurata di batterie ricaricabili, siano esse alkaline, Ni-Ca o Ni-MH. La scarica avviene con una corrente appositamente calibrata ed il micro acquisisce i dati di tensione e corrente in modo da poter ricostruire la curva caratteristica.
Il circuito è dotato anche di una porta seriale RS232 che consente la comunicazione con un PC attraverso un programma di emulazione di terminale. Le impostazioni della porta devono essere: 9600-8-N-1 senza alcun controllo di flusso. All'accensione il circuito trasmette sulla seriale il testo

*iPowerup<CR> che si rivela utile per individuare eventuali interruzioni dell'alimentazione durante il processo di scarica.
Tutti i comandi inviati al circuito attraverso il terminale devono iniziare con il carattere * seguito dal comando vero e proprio. I comandi possibili sono i seguenti:

- **i** informazioni sulla versione di firmware
- **v** lettura della tensione (valore da 0000 a 1023)
- **t** lettura della temperatura (valore da 0000 a 1023)
- **r** lettura del valore PWM (valore da 0000 a 1023)
- **p** impostazione del PWM (valore da 0000 a 1023)
- **w** scrittura del puntatore ee_adr (valore da 0000 a 0255)
- **d** scrittura del dato ee_data alla locazione puntata dal puntatore e ee_adr++ (valore da 0000 a 0255)
- **q** lettura del dato ee_data alla locazione puntata dal puntatore e ee_adr++ (valore da 0000 a 0255)
- **I** abilitazione della corrente di scarica
- **O** disabilitazione della corrente di scarica

Ad esempio inviando il comando **i* il circuito risponde con **iV1.01 10 November 2008 OZ2CPU<CR>.* ■

Compilatore
Bascom AVR

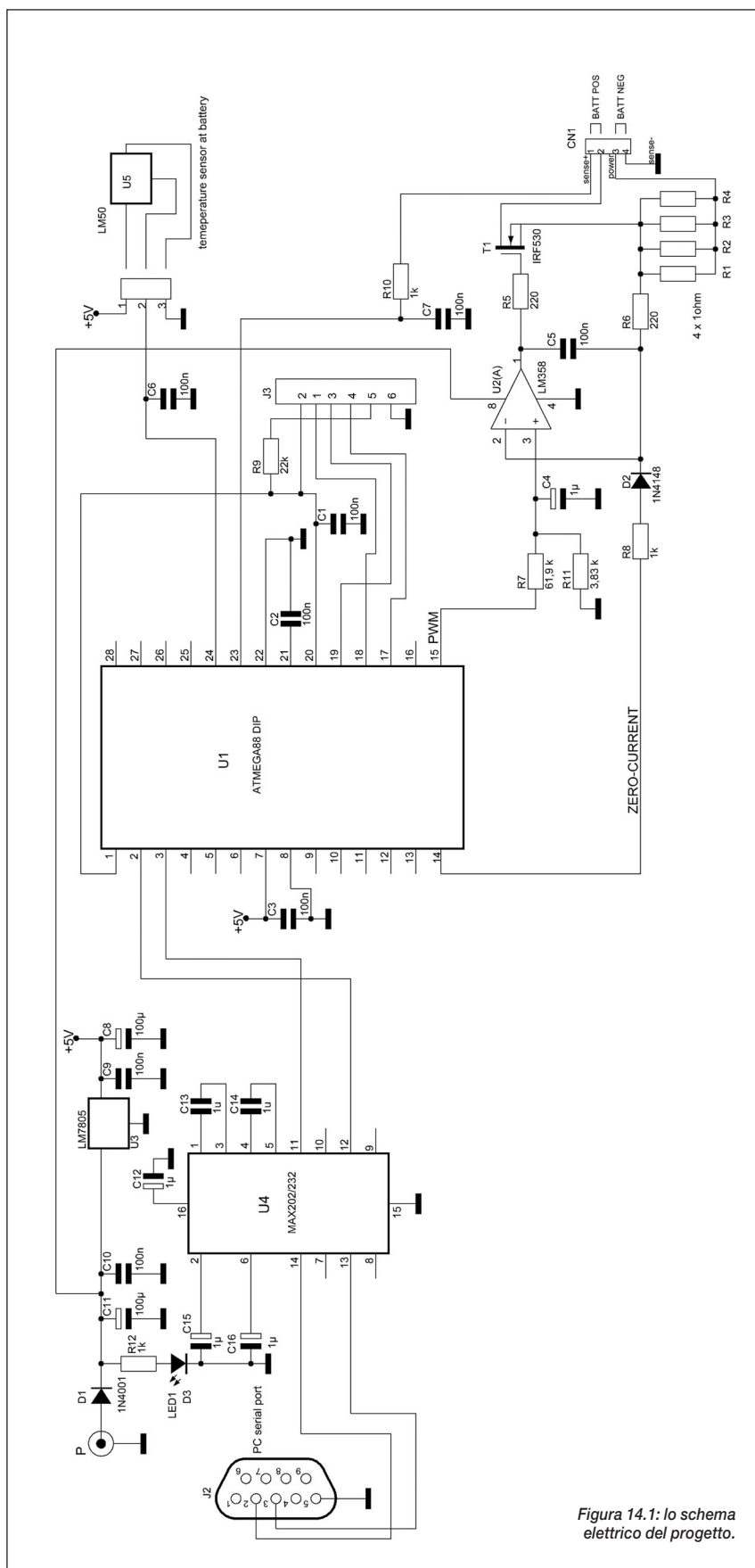



Figura 14.1: lo schema elettrico del progetto.

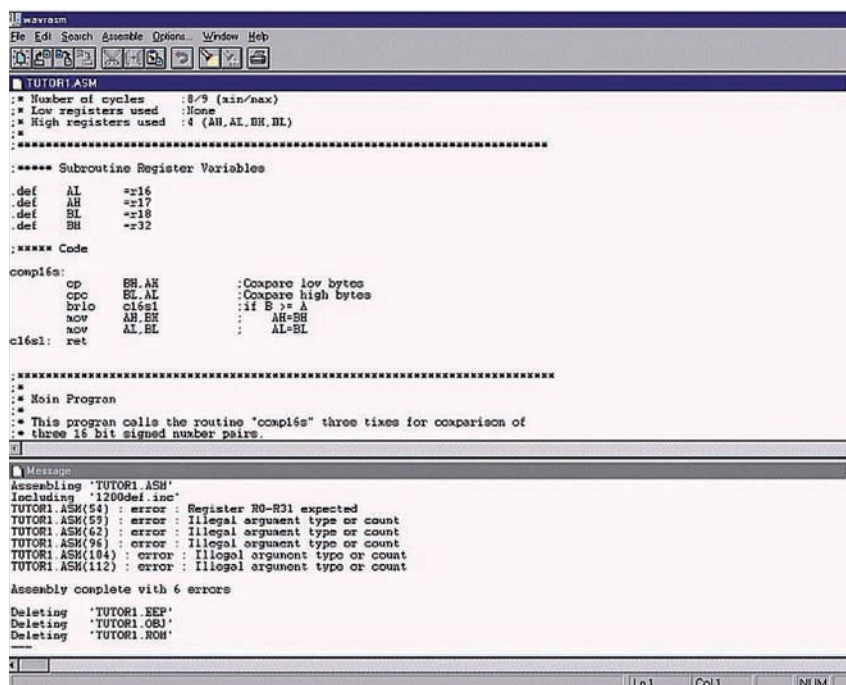
LISTA COMPONENTI

C1	= 100n
C2	= 100n
C3	= 100n
C4	= 1µ/25V
C5	= 100n
C6	= 100n
C7	= 100n
C8	= 100µ/25V
C9	= 100n
C10	= 100n
C11	= 100µ/25V
C12	= 1µ/25V
C13	= 1u/25V
C14	= 1u/25V
C15	= 1µ/25V
C16	= 1µ/25V
D1	= 1N4001
D2	= 1N4148
D3	= LED 5mm
J2	= connettore DB9 (femmina)
J3	= connettore ICD-6
P	= Connettore RCA
R1	= 1ohm 1% 1/4W
R2	= 1ohm 1% 1/4W
R3	= 1ohm 1% 1/4W
R4	= 1ohm 1% 1/4W
R5	= 220 ohm 1/4W
R6	= 220 ohm 1/4W
R7	= 61,9 Kohm 1/4W
R8	= 1 Kohm 1/4W
R9	= 22 Kohm 1/4W
R10	= 1 Kohm 1/4W
R11	= 3,83 Kohm 1/4W
R12	= 1 Kohm 1/4W
T1	= IRF530
U1	= ATMEGA88 DIP
U3	= LM7805
U4	= MAX202/232
U5	= LM50
U2(A)	= LM358

LAMPADA DA CAMPEGGIO a LED

Una modifica ad una lampada da campeggio commerciale per sostituire la lampada ad incandescenza con una più efficiente a LED della quale è possibile variare anche il grado di luminosità

L'unità da campeggio utilizzata, è alimentata da una batteria da 12V 12 Amp/ora.

L'unità è formata da:

- 12V 12 Amp/ora
- 240V powered battery charger
- uno strumento per la misura dei 12V
- una lampada ad incandescenza
- una presa a 12V per alimentare dispositivi esterni due morsetti per collegarlo alla batteria dell'auto

Questo sistema viene utilizzato per alimentare dispositivi a 12V (lettori DVD, ecc.), come luce d'emergenza e come start per l'auto. La lampada ad incandescenza da 8W non è molto luminosa e consuma troppo. L'idea, dunque è quella di sostituirla con una lampada a LED pilotata da una scheda a microcontrollore che ne regola la luminosità. La lampada originale da 8W è stata sostituita con 9 LED bianchi da 20.000 mCandele a 20mA.

Questa lampada a LED fornisce una qualità di luce migliore consumando da un minimo di 15mA ad un massimo di 60mA a 12V.

Questo significa che pilotando i LED con 15 mA si ha una potenza pari a: P = V x I = 12 x 0,015 = 0,18W.

Figura 4: AVR assembler

pilotando i LED con 60mA si ha invece:
P = 12 x 0,06 = 0,72W

Il microcontrollore usato

Il microcontrollore utilizzato è l' 8 pin Atmel Tiny15, che ha le seguenti caratteristiche:

- 1K flash (program) memory
- 32 registri usabili come RAM
- 64 bytes EEPROM
- 4 canali a 10 bit ADC
- fino a 6 linee I/O
- oscillatore interno a 1.6MHz

Scheda LED

Il circuito ha 9 LED pilotati in serie a gruppi di 3. Con una tensione di accensione di circa 3.3V per LED dovremmo usare circa 9.9V uper accendere ogni gruppo di 3. Per limitare la corrente a 20 mA per ogni gruppo di LED è stata messa una resistenza in serie.
Usando una resistenza da 100 Ohm sulla scheda micro, tenendo conto della resistenza del BSN254A E FET (5 Ohm), abbiamo 105 totali in serie ad ogni gruppo.

Scheda di controllo

La scheda di controllo utilizza una Tiny15 alimentato da un 78L05, questo regolatore eroga 100mA a 5V (i LED sono pilotati direttamente dai FET e non dal 78L05).
Il tiny15 controlla ogni gruppo di LED tramite I segnali della porta PORTB pilotando il gate dei FET:
- PB0 – LED gruppo 1 (FET Q1);
- PB1 – LED gruppo2 (FET Q2);
- PB2 – LED gruppo 3 (FET Q3).
Ognuno di questi segnali piloterà un FET (Q1 to Q3) che controllerà l'accensione di ogni gruppo di LED.

Quando il segnale della PORTB è LOW, il gruppo di LED sarà spento. Quando il segnale della PORTB sarà

Segue a pagina 60

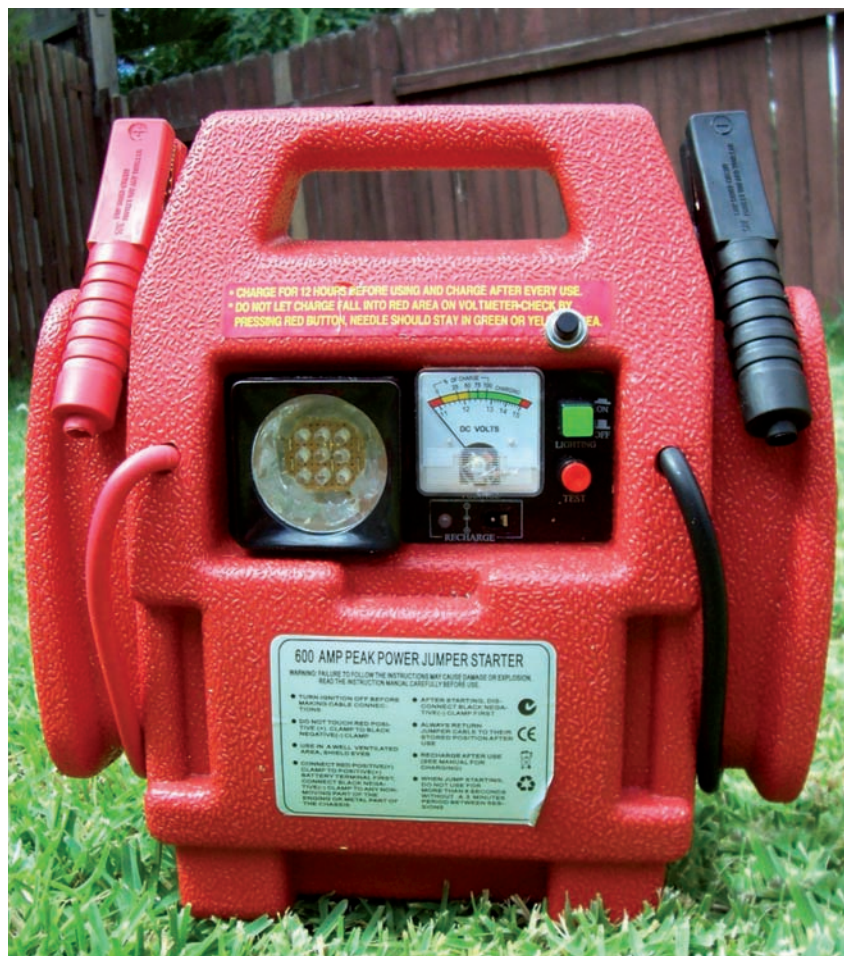



Figura 15.5: unità da campeggio power start.

TABELLA 1: CORRENTE CONSUMATA DAI 12V

LUMINOSITÀ	CORRENTE CONSUMATA	LED GRUPPO 1	LED GRUPPO 2	LED GRUPPO 3
1	14.3 mA	on 33%	off	off
2	20.4 mA	on 66%	off	off
3	26.7 mA	on 100%	off	off
4	32.4 mA	on 100%	on 33%	off
5	38.2 mA	on 100%	on 66%	off
6	44.2 mA	on 100%	on 100%	off
7	49.6 mA	on 100%	on 100%	on 33%
8	55.0 mA	on 100%	on 100%	on 66%
9	60.76 mA	on 100%	on 100%	on 100%

```
; register definitions
.def  status = R2

.def  temp = R16            ; general temporary
                            ; register
.def  tempa = R17           ; used in TIM0_OVF
.def  tempb = R18           ; second temporary
                            ; general register
.def  brightness = R19      ; 1 to 9
.def  cycle_cnt = R20       ; 1 to 3
.def  del_cnt = R21         ; dec'd in TIM0_OVF
.equ  CYCLE = 3             ; 1 to 3
.equ  MAX = 10              ; Brightness
.equ  FREQ = 0xAD           ; 3.3333mSec
.equ  SWITCH = 3            ; PB3
.cseg
;————————————————
;Reset handler
  .org $0000
  rjmp  STARTUP
;————————————————
; INT0
  reti
;————————————————
; Pin Change
  reti
;————————————————
; Timer1 Compare
  reti
;————————————————
;Timer 1 Overflow
  reti
;————————————————
;Timer0 overflow handler
  rjmp  TIM0_OVF
;————————————————
; Eprom ready
  reti
;————————————————
;Analogue Compare
  reti
```

```
;————————————————
;ADC Conversion
  reti
;—————————————————-
; TIMER0 Overflow Interrupt
; dec cycle_cnt and del_cnt till zero
;—————————————————-
;
TIM0_OVF:
  in status,SREG            ; save the status
                            ; register at entry
  ldi  tempa,FREQ
  out  TCNT0,tempa          ; restart TIMER0
; if del_cnt is non-zero, dec till 0
  tst  del_cnt              ; is del_cnt non-zero?
  breq  TIM_01              ; no!
  dec  del_cnt              ; yes so dec it
                            ; cycle_cnt goes
                            ; from 1 to 3
TIM_01:
  inc  cycle_cnt            ; inc cycle_cnt
  cpi  cycle_cnt,CYCLE+1    ; wrap around?
  brne  TIM_02              ; no!
  ldi  cycle_cnt,1          ; yes! Reset to 1
TIM_02:
  out  SREG,status          ; restore status
                            ; register at exit
  reti


;————————————————
; LED lookup table
;————————————————
LOOKUP:
  .DB
0x01,0x00,0x00,0x01,0x01,0x00,0x01,0x01,0x01,0
x03,0x01,0x01,0x03,0x03,0x01,0x03,0x03,0x03,0x
07,0x03,0x03,0x07,0x07,0x03,0x07,0x07,0x07
;————————————————
; This is where we setup the ports
; and peripherals
;————————————————
```

```
;
STARTUP:
; PB0,PB1,PB2 are outputs
   ldi  temp,0x07
   out  DDRB,temp
; Pull-ups enabled PB3 for SW1
   ldi  temp,0x08
   out  PORTB,temp
; set up internal oscillator
   ldi  temp,0x60
   out  OSCCAL,temp
; CLK/64 3.333 mSec clock
   ldi  temp,0x03
   out  TCCR0,temp
   ldi  temp,0
   out  TCCR1,temp
; Timer 0 Overflow interrupts 3.3 mSec
   ldi  temp,(1<<TOIE0)
   out  TIMSK,temp
; set up timer0
   ldi  temp,FREQ
   out  TCNT0,temp
; set up initisl variables
   ldi  cycle_cnt,1
   ldi  brightness,1
   sei                        ; enable interrupts
;————————————————————
; This is where the program loops
;————————————————————
;
A_LOOP:
   tst  del_cnt               ; are we delaying?
   brne  A_LOOP10             ; yes!
                              ; So no switch checking!
   sbic  PINB,SWITCH          ; no! Is button
                              ; pressed?
   rjmp  A_LOOP10             ; no!
   inc  brightness            ; yes! So inc brightness
   ldi  del_cnt,100           ; don't check button
                              ; for 330 mSec
   cpi  brightness,MAX        ; wrap around?
   brne  A_LOOP10             ; no!
   ldi  brightness,1          ; yes!
; We will set the pointer to the start of the
; appropriate '1 of 3' byte patterns
; in the table LOOKUP. Then we will add the
; cycle_cnt number to the pointer
; to point to the appropriate display pattern.
A_LOOP10:
   ldi  ZH, high(LOOKUP<<1)     ; Initialize Z
                                ; pointer to LOOKUP
   ldi  ZL, low(LOOKUP<<1)
   mov  temp,brightness
A_LOOP12:
   dec  temp                    ; allow for 1 origin
   breq  A_LOOP15               ; if eq 0 1st time
                                ; then at start of table
   ldi  tempb,3                 ; else add 3
     ; (next brightness level)
   add  ZL,tempb
   rjmp  A_LOOP12               ; back for more
A_LOOP15:
   add  temp,cycle_cnt        ; offset to cycle_cnt byte
   dec  temp                  ; allow for 1 origin
   add  ZL,temp               ; now pointing at
                              ; LED pattern
   lpm                        ; R0 has LED pattern
   ldi  temp,0x08
   or  R0,temp                ; make sure SWITCH
                              ; pullup is high
   out  PORTB,R0              ; O/P LED state
   rjmp  A_LOOPListato 1
rem Utilizzo dei pesi
rem per ottenere 3,3 Volt

program dac
portb=0                       'Azzera portb
trisb=0                       'definisce PORTB in output
portb=171                     'Imposta la PORTB con i
     'bit utili per ottenere 3,3V
     'oppure 10101011 in binario
end.
```

TABELLA 2: LUMINOSITÀ LAMPADA LED IN FUNZIONE DEL NUMERO DI PRESSIONI DEL PULSANTE

PRESSIONE PULSANTE	LED GRUPPO 1	LED GRUPPO 2	LED GRUPPO 3
All'accensione (default)	33%	0%	0%
1	66%	0%	0%
2	100%	0%	0%
3	100%	33%	0%
4	100%	66%	0%
5	100%	100%	0%
6	100%	100%	33%
7	100%	100%	66%
8	100%	100%	100%
9	33%	0%	0%

HIGH il gruppo di LED sarà acceso. Il pulsante normalmente aperto sarà collegato tra GND e PB3. La capacità del Tiny15 di collegare un pull-up resistor (via software) ai 5V servirà per capire quando il pulsante verrà premuto.
La **tabella 1** mostra l'assorbimento di corrente dalla batteria a 12V per i 9 diversi livelli di luminosità disponibili (tenete presente che il microcontrollore usa circa 6 o 7 mA a 5V).

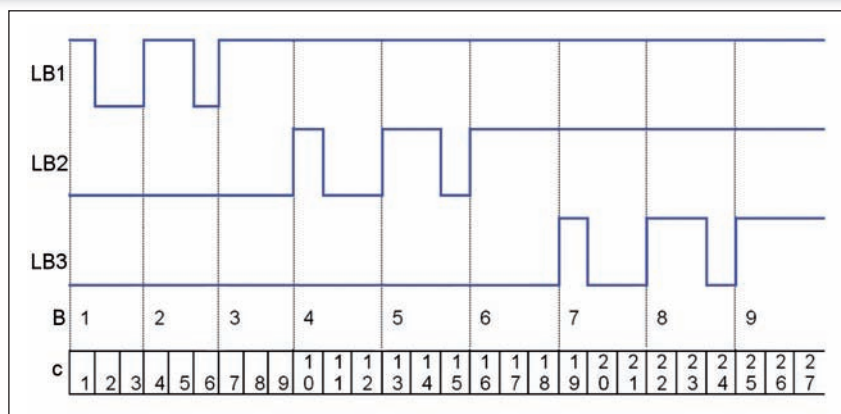



Figura 15.2: cicli di luminosità minimo e massimo.

Assemblaggio e test

Il progetto è formato da tre parti da montare all'interno della lampada:
1) la scheda LED;
2) la scheda microcontrollore;
3) il pulsante.
Si sostituisce la lampada ad incandescenza con quella a LED, si collega la scheda microcontrollore alla lampada LED ed alla batteria e si collega uno dei contatti del pulsante a massa. Dopo aver assemblato il tutto possiamo accendere l'unità ed iniziare con il test. Premendo la prima volta il pulsante della lampada (quello verde) I LED si accenderanno al livello 1 di luminosità (33% del tempo accesi).
Con questo basso livello di consumo si può lasciare accesa la lampada anche tutta la notte utilizzandola come lamdada di servizio.
Ogni volta che premo il pulsante la luminosità aumenterà di livello.
Quando arriveremo al livello 9, la lu-

Figura 15.6: lampada a LED.

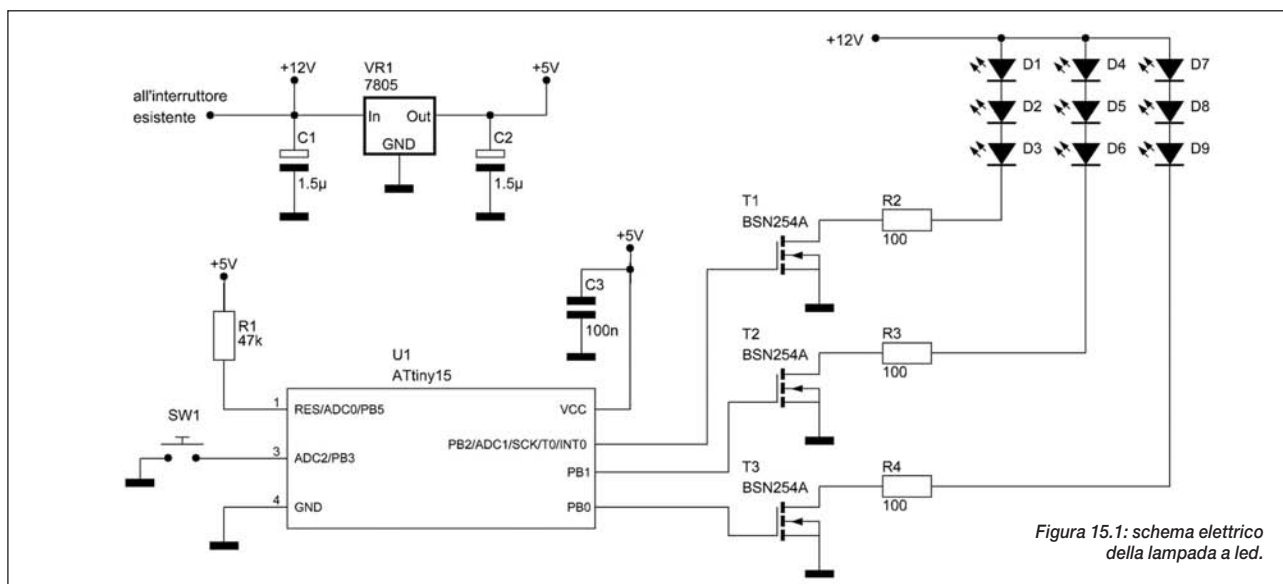



Figura 15.1: schema elettrico della lampada a led.

LISTA COMPONENTI

C1	= 1.5µ	R2	= 100	T2	= BSN254A
C2	= 1.5µ	R3	= 100	T3	= BSN254A
C3	= 100n	R4	= 100	U1	= ATtiny15
D1-D9	= LED 20.000mCandela 20mA	SW1	= pulsante n.a.	VR1	= 7805
R1	= 47k	T1	= BSN254A		

minosità sarà sufficiente per leggere un libro, con maggior efficienza rispetto all'originale. Si può lasciare il sistema acceso per 4 giorni a piena potenza (massima luminosità) senza problemi.

Figura 15.7: particolare del montaggio.

Descrizione del programma

Il programma è scritto in AVR assembler usando AVR studio versione 4.13. I vettori di Interrupt sono definiti all'inizio del codice. Ci sono solo due vettori utilizzati:
STARTUP and TIMO_OVF, ma per sicurezza tutti i vettori sono stati definiti con le opportune RETI.
Il Timer0 overflow interrupt TIMO_OVF è stato impostato per lanciare un interrupt ogni 3.3333 mSec che si occuperà di:

• decrementare delay_cnt fino a zero
• incrementare cycle_cnt (da 1 a 3)

La pressione del pulsante SW1 incrementerà il valore della variabile brightness in un range che va da 1 (minimum light) a 9 (maximum light). Quando il pulsante viene premuto velocemente la variabile brightness verrà incrementata e viene impostato un tempo di ritardo (333.33 mSec di ritardo settando delay_cnt a 100) prima di riconoscere una nuova pressione del tasto per evitare problemi di rimbalzo.
Un ciclo completo di accensione per i gruppi di LED occupa 3 cicli di 3.3333 mSEC, così un ciclo completo per la lampada LED è di circa 10 mS. Questo permette di avere un ciclo di refresh di circa 100Hz per ogni gruppo di LED che evita di vedere sfarfallamenti.
La tabella LOOKUP contiene le impostazioni per ogni gruppo di LED per i 9 livelli di luminosità. A_LOOP usa le variabili brightness e cycle_count per caricare i valori appropriati per ogni gruppo di LED e portarli all'uscita PORTB. Come si vede in **tabella 2**, ogni valore di luminosità B controllerà ogni gruppo di LED (LB1 to LB3) a occuperà 3 cicli C di 3.3333 mSec per ogni livello di luminosità.
La **figura 15.2** mostra le forme d'onda risultatanti dal pilotaggio dei 3 gruppi di LED (LB1 to LB3) dopo le varie pressioni del pulsante con la conseguente variazione di luminosità B. ■

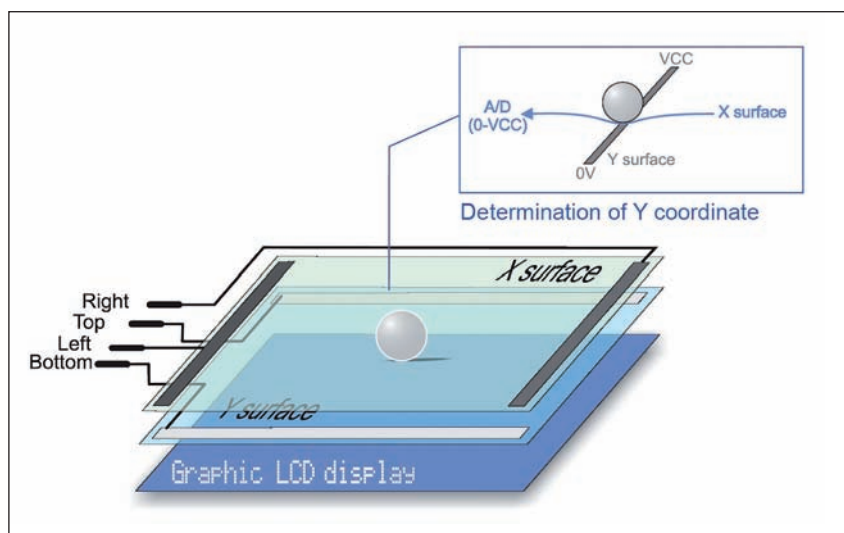
AVR TOUCHSCREEN

Gestire un display grafico touchscreen con un ATmega16. L'hardware ed il software

E' possibile dotare qualunque display grafico della funzionalità touch-screen semplicemente aggiungendo al display una menbrana touch. Esistono diversi tipi di membrane. La più semplice è quella resistiva costituita da due fogli trasparenti X e Y disposti a formare una struttura a sandwitch all'interno della quale si trova uno strato resistivo. Ai lati dei due fogli sono praticate delle metallizzazioni che fanno capo ai terminali di uscita **(figura 16.1)**. Il processo mediante il quale si risale alle coordinate del punto in cui viene effettuata la pressione sulla membrana può essere riassunto in due passi principali. Il primo passo è la determinazione della coordinata X mentre il secondo passo consiste nella determinazione della coordinata Y. Per la determinazione della coordinata X si pone a massa il contatto di sinistra della membrana X e si collega al positivo di alimentazione il contatto di destra. Si ottiene quindi un partitore resistivo il cui rapporto di partizione dipende dal punto che viene premuto. La tensione di uscita viene letta sul terminale Bottom della membrana Y e può assumere valori compresi tra 0 e +Vcc. Più il punto di pressione è vicino al contatto di sinistra della membrana X più la tensione di uscita si avvicina allo zero. Per la determinazione della coordinata Y il procedimento è analogo: si collegano i contatti Top e Bottom della membrana Y rispettivamente a +Vcc e massa e si legge la tensione di uscita dal terminale di sinistra della membrana X.

Per gestire un touchscreen con un microcontrollore (nel nostro caso un ATmega16) è necessario un semplice hardware per la gestione dei contatti

segue a pagina 65

Figura 16.1: il principio di funzionamento di un touch screen resistivo e la determinazione della coordinata Y.

LISTA COMPONENTI

C1	= 100n	IC1	= ATmega16	R6	= 47k	R14	= 1M
C2	= 100n	IC2	= 74HC04	R7	= 1k	R15	= 1k
C3	= 22p	P1	= 10k trimmer	R8	= 1k	T1	= BC546
C4	= 22p	R1	= 1k	R9	= 47k	T2	= BC556
C5	= 100n	R2	= 1k	R10	= 1k	T3	= BC556
D1	= LED 5mm	R3	= 1k	R11	= 100	T4	= BC546
D2	= LED 5mm	R4	= 1k	R12	= 1k	T5	= BC546
DIS1	= Display grafico 128x64	R5	= 1k	R13	= 1k	X1	= Quarzo 8MHz

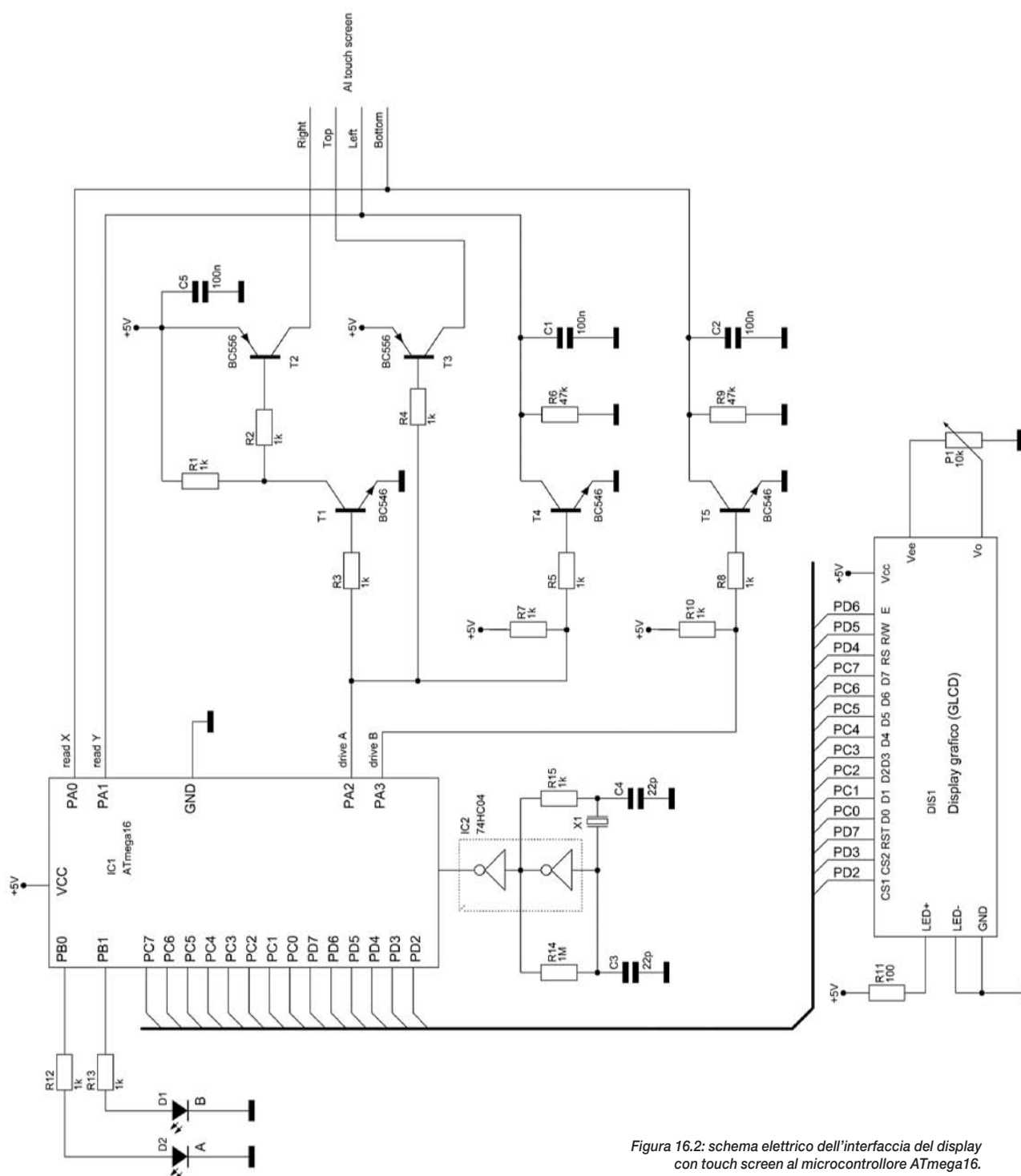



Figura 16.2: schema elettrico dell'interfaccia del display con touch screen al microcontrollore ATmega16.

LISTATO 1

```
char GLCD_DataPort at PORTC; char GLCD_DataPort_Direction at DDRC;
sbit GLCD_CS1 at PORTD.B2; sbit GLCD_CS1_Direction at DDRD.B2;
sbit GLCD_CS2 at PORTD.B3; sbit GLCD_CS2_Direction at DDRD.B3;
sbit GLCD_RS at PORTD.B4; sbit GLCD_RS_Direction at DDRD.B4;
sbit GLCD_RW at PORTD.B5; sbit GLCD_RW_Direction at DDRD.B5;
sbit GLCD_EN at PORTD.B6; sbit GLCD_EN_Direction at DDRD.B6;
sbit GLCD_RST at PORTD.B7; sbit GLCD_RST_Direction at DDRD.B7; // End Glcd module connections
sbit DRIVE_A at PORTA.B2; sbit DRIVE_A_Direction at DDRA.B2; // Touch Panel module connections
sbit DRIVE_B at PORTA.B3; sbit DRIVE_B_Direction at DDRA.B3; // End Touch Panel module connections
long x_coord, y_coord, x_coord128, y_coord64; // scaled x-y position

unsigned int GetX() { //reading X
        DRIVE_A = 1; // DRIVEA = 1 (LEFT drive on, RIGHT drive on, TOP drive off )
        DRIVE_B = 0; // DRIVEB = 0 (BOTTOM drive off )
        Delay_ms(5);
        return ADC_Read(0); // READ-X (BOTTOM)
}

unsigned int GetY() { //reading Y
        DRIVE_A = 0; // DRIVEA = 0 (LEFT drive off , RIGHT drive off , TOP drive on)
        DRIVE_B = 1; // DRIVEB = 1 (BOTTOM drive on)
        Delay_ms(5);
        return ADC_Read(1); // READ-X (LEFT)
}

void main() {
        DRIVE_A_Direction = 1; // Set DRIVE_A pin as output
        DRIVE_B_Direction = 1; // Set DRIVE_B pin as output
        PORTB.B0 = 0;
        DDRB.B0 = 1; // Set PB0 pin as output (Default value 0)
        PORTB.B1 = 0;
        DDRB.B1 = 1; // Set PB1 pin as output (Default value 0)
        Glcd_Init(); // Initialize GLCD
        Glcd_Fill(0); // Clear GLCD
        Glcd_Set_Font(font5x7, 5, 7, 32); // Choose font,
        Glcd_Fill(0);
        Glcd_Write_Text(“Touchscreen EXAMPLE”,10,0,1);
        Glcd_Write_Text(“MIKROELEKTRONIKA”,17,7,1);
        Glcd_Rectangle(8,16,60,48,1); //Display Buttons on GLCD:
        Glcd_Rectangle(68,16,120,48,1);
        Glcd_Box(10,18,58,46,1);
        Glcd_Box(70,18,118,46,1);
        Glcd_Write_Text(“BUTTON1”,14,3,0);
        Glcd_Write_Text(“PB0 OFF”,14,4,0);
        Glcd_Write_Text(“BUTTON2”,74,3,0);
        Glcd_Write_Text(“PB1 OFF”,74,4,0);
        while (1) { // read X-Y and convert it to 128x64 space
                x_coord = GetX();
                y_coord = GetY();
                x_coord128 = (x_coord * 128) / 1024;
                y_coord64 = 64 -((y_coord *64) / 1024);
                //if BUTTON1 is selected
                if ((x_coord128 >= 10) && (x_coord128 <= 58) && (y_coord64 >= 18) && (y_coord64 <= 46)) {
                        if(PORTB.B0 == 0) {
                                PORTB.B0 = 1;
                                Glcd_Write_Text(“PB0 ON “,14,4,0);
                        }
                        else {
                                PORTB.B0 = 0;
                                Glcd_Write_Text(“PB0 OFF”,14,4,0);
                                }
                }
                //if BUTTON2 is selected
                if ((x_coord128 >= 70) && (x_coord128 <= 118) && (y_coord64 >= 18) && (y_coord64 <= 46)) {
                        if(PORTB.B1 == 0) {
                                PORTB.B1 = 1;
                                Glcd_Write_Text(“PB1 ON “,74,4,0);
                                }
                        else {
                                PORTB.B1 = 0;
                                Glcd_Write_Text(“PB1 OFF”,74,4,0);
                                }
                }
                Delay_ms(100);
        }
}
```

della membrana touch in accordo a quanto appena descritto. Lo schema di riferimento è quello di **figura 16.2**. Il contatto Bottom della faccia Y ed il contatto Sinistra della faccia X sono connessi agli ingressi del convertitore AD e le coordinate X ed Y vengono determinate leggendo la tensione a questi due terminali.

Dal punto di vista del software è necessario dapprima costruire un menu per l'interfaccia utente, applicare le tensioni ai contatti della membrana touch, quindi leggere le tensioni di uscita per determinare quale sia la zona premuta dall'utente. Una volta determinate le coordinate è possibile decidere quale azione intraprendere in base alla scelta effettuata dall'utente. Come esempio pratico si può far riferimento al **listato 1** il quale descrive come portare a livello alto o livello basso due pin del micro (collegati a due led A e B) a seconda del pulsante premuto sullo schermo del display. ■

IMPORTANTE: NON SCRIVERE NELLA ZONA SOPRASTANTE!

AVVERTENZE

Il Bollettino deve essere compilato in ogni sua parte (con inchiostro nero o blu) e non deve recare abrasioni, correzioni o cancellature.

La causale è obbligatoria per i versamenti a favore delle Pubbliche Amministrazioni.

Le informazioni richieste vanno riportate in modo identico in ciascuna delle parti di cui si compone il bollettino.

DIRETTORE RESPONSABILE
Antonio Cirella

DIRETTORE TECNICO
Maurizio Del Corso

Comitato Scientifico
Simone Masoni (Microtest), Francesco Picchi (Microtest), Massimo Rovini (Università degli Studi di Pisa), Tiziano Galizia (Tigal), Claudio Turchetti (Università Politecnica delle Marche).

Segreteria di redazione
Fabiana Rosella

Art Director
Patrizia Villa

Direzione Redazione Pubblicitá International Advertisement
INWARE Edizioni srl
Via Cadorna, 27/31
20032 Cormano (MI)
Tel. 02.66504755
Fax 02.66508225
info@inwaredizioni.it
www.inwaredizioni.it
Redazione: fe@inwaredizioni.it

Stampa
Tiber S.p.a.
Via della Volta, 179
25124 Brescia (Italy)

Distribuzione
Parrini & C. S.p.a.
Viale Forlanini, 23
20134 Milano

Ufficio Abbonamenti
INWARE Edizioni srl
Via Cadorna, 27/31
20032 Cormano (MI)
Per informazioni, sottoscrizione o rinnovo dell'abbonamento:
abbonamenti@inwaredizioni.it
Tel. 02.66504755
Fax. 02.66508225
L'ufficio abbonamenti è disponibile telefonicamente dal lunedì al venerdì dalle 14,30 alle 17,30.
Tel. 02.66504755
Fax 02.66508225

Abbonamento per l'Italia:
€ 49,50
Abbonamento per l'estero:
€ 115,00
Gli arretrati potranno essere richiesti, per iscritto, a **€ 9,00** oltre le spese di spedizione

Autorizzazione alla pubblicazione
Tribunale di Milano n. 601
del 10/10/2008

Privacy
Nel caso la rivista sia pervenuta in abbonamento o in omaggio, si rende noto che i dati in nostro possesso sono impiegati nel pieno rispetto del D.Lgs. 196/2003. I dati trasmessi a mezzo cartoline o questionari presenti nella rivista, potranno venire utilizzati per indagini di mercato, proposte commerciali, o l'inoltro di altri prodotti editoriali a scopo di saggio. L'interessato potrà avvalersi dei diritti previsti dalla succitata legge. In conformità a quanto disposto dal Codice di deontologia relativo al Trattamento di dati personali art. 2, comma 2, si comunica che presso la nostra sede di Cormano Via Cadorna 27, esiste una banca dati di uso redazionale. Gli interessati potranno esercitare i diritti previsti dal D.Lgs. 196/2003 contattando il Responsabile del Trattamento Inware Edizioni Srl (info@inwaredizioni.it).

Collaborare con FARE ELETTRONICA
Le richieste di collaborazione vanno indirizzate all'attenzione di Maurizio Del Corso (m.delcorso@inwaredizioni.it) e accompagnate, se possibile, da una breve descrizione delle vostre competenze tecniche e/o editoriali, oltre che da un elenco degli argomenti e/o progetti che desiderate proporre.